AI&DS

Interim Report
AI&DS TR-1048-02

July 1984

DISTRIBUTED HYPOTHESIS TESTING IN DISTRIBUTED SENSOR NETWORKS

INTERIM REPORT

Prepared for:

Dr. Barry M. Leiner
DARPA
1400 Wilson Boulevard
Arlington, VA 22209

Prepared by:

C.Y. Chong
S. Mori
M.R. Fehling
K.C. Chang
R.P. Wishner

Approved for public release: distribution unlimited.

ADVANCED INFORMATION & DECISION SYSTEMS
Mountain View, CA 94040

AI&DS
ADVANCED INFORMATION
& DECISION SYSTEMS

201 San Antonio Circle, Suite 286
Mountain View, CA 94040
(415) 941-3912

Interim Report
AI&DS TR-1048-02

July 1984

DISTRIBUTED HYPOTHESIS TESTING IN DISTRIBUTED SENSOR NETWORKS

INTERIM REPORT

C.Y. Chong
S. Mori
M.R. Fehling
K.C. Chang
R.P. Wishner

Sponsored by

Defense Advanced Research Projects Agency (DoD)
DARPA Order No. 4272/4

Under Contract No. MDA903-83-C-0372 issued by Department of Army, Defense Supply Service - Washington, Washington, D.C., 20310

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AIDS TR-1048-02	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Advanced Information & Decision Systems 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Defense Supply Service - Washington
6c. ADDRESS (City, State and ZIP Code) 201 San Antonio Circle, Suite 286 Mountain View, CA 94040	7b. ADDRESS (City, State and ZIP Code) Room 1D 245, The Pentagon Washington, DC 20310
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency 8b. OFFICE SYMBOL (If applicable) IPTO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-83-C-0372
8c. ADDRESS (City, State and ZIP Code) 1400 Wilson Boulevard Arlington, VA 22209	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Distributed Hypothesis Testing in Distributed Sensor Networks	
12. PERSONAL AUTHOR(S) Chong, C.Y., Mori, S., Fehling, M.R., Chang, K.C., Wishner, R.P.	
13a. TYPE OF REPORT INTERIM 13b. TIME COVERED FROM 7/15/83 TO 7/14/84	14. DATE OF REPORT (Yr., Mo., Day) 1984 July 15. PAGE COUNT 154
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Distributed Sensor networks, distributed estimation, distributed multitarget tracking, information fusion, group tracking, distributed system testbeddesign, testbed architecture

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report presents research results on distributed situation assessment in distributed sensor networks (DSNs). Information fusion algorithms for hypothesis formation and evaluation are presented. The algorithms are based on the concept of an information graph. A theory for tracking groups of targets has been developed based on a Bayesian Theory for multitarget tracking. Finally the representation of knowledge and architecture for a testbed to design and test a DSN are discussed.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

LIST OF FIGURES

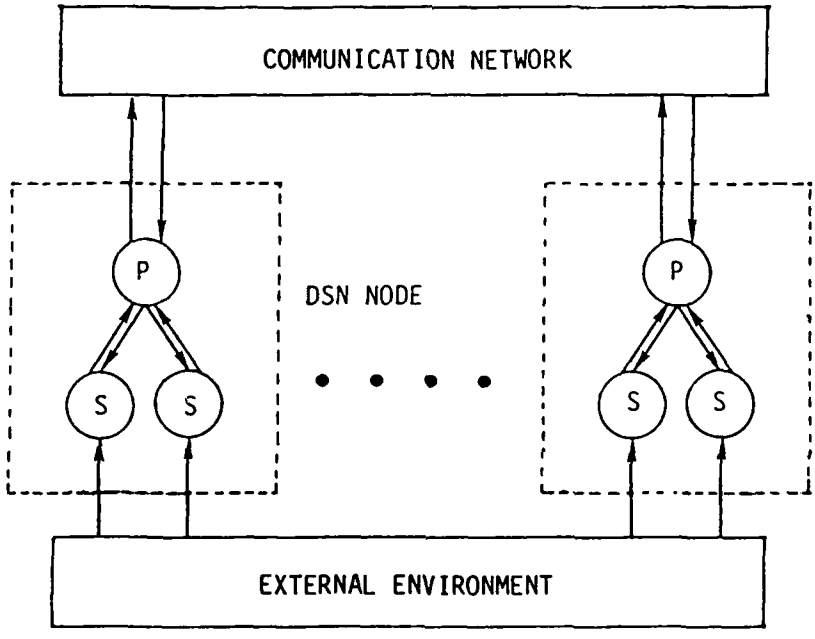
1. INTRODUCTION AND SUMMARY

This interim technical report describes research performed on the distributed processing of sensor data for situation assessment in a distributed sensor network (DSN). This research has been performed at Advanced Information & Decision Systems under the contract entitled "Distributed Hypothesis Testing in Distributed Sensor Networks".

1.1 DSN PROBLEM DESCRIPTION

We assume a system structure as in Figure 1-1. There is a system of distributed sensor/processor nodes. Each node may have one or more sensor types, and the sensors from different nodes may have overlapping coverage. The sensors collect data from the environment and pass them on to the processors (processing nodes). The processing nodes process the sensor data and communicate with other nodes through the communication network to obtain an assessment of the state of the world. It is generally assumed that no single node possesses complete information and each node may have a different world model. The processing nodes can also control the sensors to improve on the performance of the overall system.

A distributed sensor network can be used for many applications. We are particularly interested in a DSN which is used for the tracking and classification of multiple targets. The target environment is assumed to be dense, so that determining the origins of the measurements in a particular sensor report is not always easy. The problem is further complicated by the presence of false alarms and missing target reports. In such a network, tracking and classification is highly dependent on identifying the right data association hypotheses. Since the nodes in general have access to different information, communication among the nodes can greatly improve the performance of the system. The problem is thus one of distributed hypothesis formation and evaluation, which we can abbreviate



S = SENSOR

P = PROCESSOR

Figure 1-1: Distributed Sensor Network

as distributed hypothesis testing.

In our previous DSN project we initiated research on the distributed tracking of multiple targets by the nodes of a distributed sensor network. In the following we shall review a model of the processing node that has been proposed.

1.2 PROCESSING NODE MODEL

The processing nodes are the main information processing units in the DSN. Each processing node collects measurements from a set of sensors. Its function is to process the local sensor data to form an assessment of the state of the world, to combine the information obtained from other nodes with the local information to update its assessment, to distribute information to other nodes, and to performs these functions effectively. These functions are performed in four separate modules within each processing node (see Figure 2-2). In the following we shall discuss the modules in more detail.

1.2.1 Generalized Tracker/Classifier

This module is responsible for the local data processing before any communication with the other nodes takes place. Since the objective of the system under consideration is the tracking and classification of multiple targets, this module is a multitarget tracker. In the previous project, we have developed a general theory for multitarget tracking which is implemented in the form of the *Generalized Tracker/Classifier* (GTC). The GTC has the structure shown in Figure 2-3 and itself consists of four modules. The *hypothesis formation* module forms multiple hypotheses from the sensor data, each consisting of a collection of tracks to explain the origins of the measurements in each data set. These hypotheses are then evaluated by the *hypothesis evaluation* module with respect to their probabilities of being true. The *filtering and parameter estimation* module generates state estimates and classifications for each track. It is essential for hypothesis evaluation and can thus be viewed as a submodule. To stay within the computational constraints of each node, the hypotheses are pruned, combined, clustered, etc. This takes place in the *hypothesis management* module.

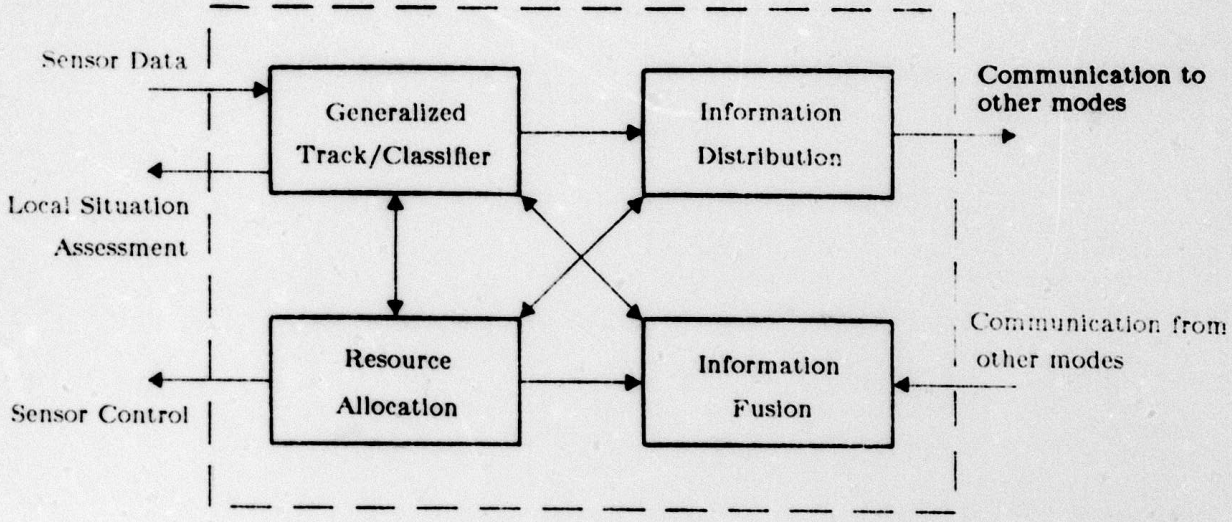


Figure 2-1: Structure of Processing Node

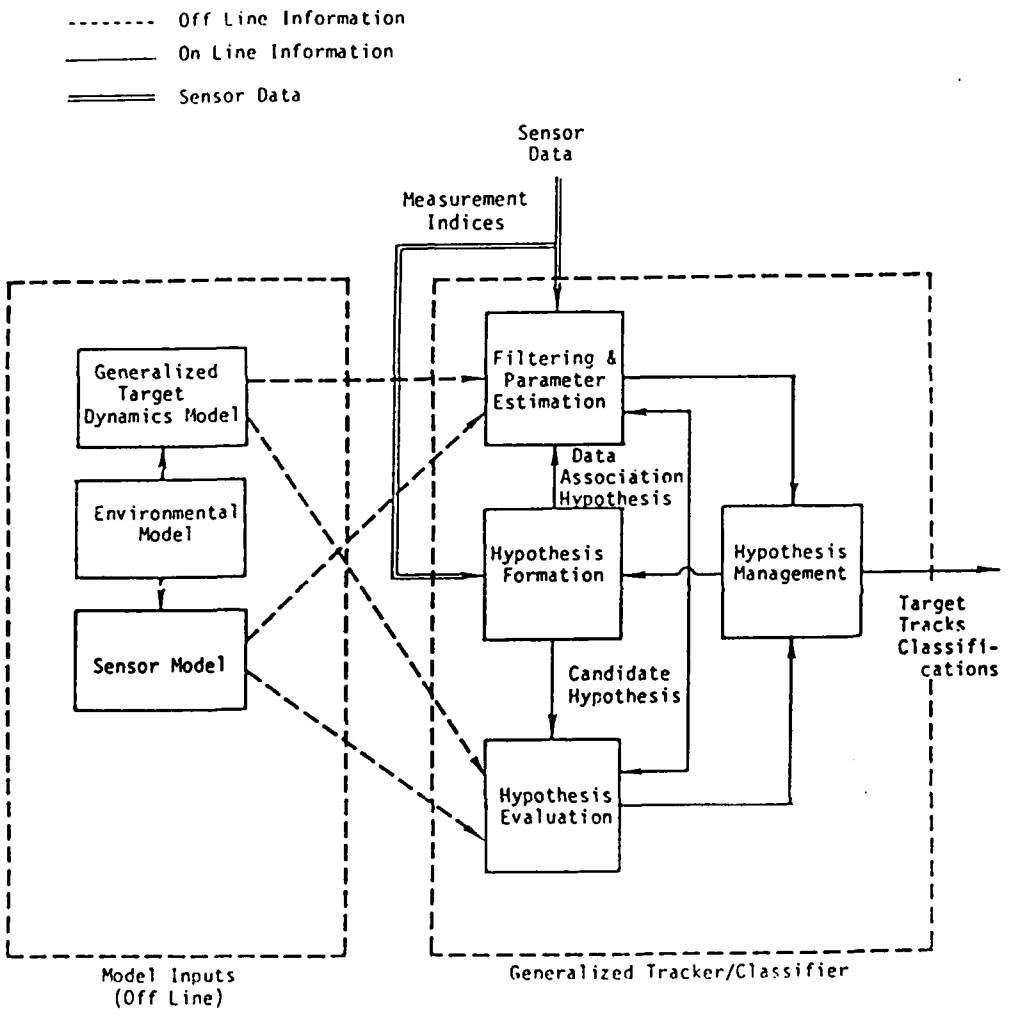


Figure 1-3: Generalized Tracker/Classifier

The result of this processing is a set of hypotheses and their probabilities, a collection of tracks corresponding to possible targets and the state distributions of these tracks. These quantities together constitute the information state for multitarget tracking.

1.2.2 Information Fusion

This module combines the local information with information obtained from the other nodes to obtain a new situation assessment. The information from the local nodes consists of the information described above. The information from other nodes is also similar. Information fusion then consists of the following steps (see Figure 2-4):

1. *Hypothesis Formation* - Given a set of hypotheses from other nodes, this submodule generates new global hypotheses. Tracks from the hypotheses of different nodes are associated in all possible ways, whether they correspond to the same or different targets.

2. *Hypothesis Evaluation* - Each of the hypotheses formed above is then evaluated with respect to its probability of being true. The statistics of the tracks from different hypotheses are used in this evaluation. For example, if two tracks are widely apart in their position or velocity distributions, they are more likely to have come from different targets than the same target.

3. *Hypothesis Management* - This is again needed to make computation feasible within the available resources.

1.2.3 Information Distribution

This module decides what information is to be transmitted, who gets the information, and when it should be communicated. It thus specifies the information available to each node at any time, i.e., the information structure of the

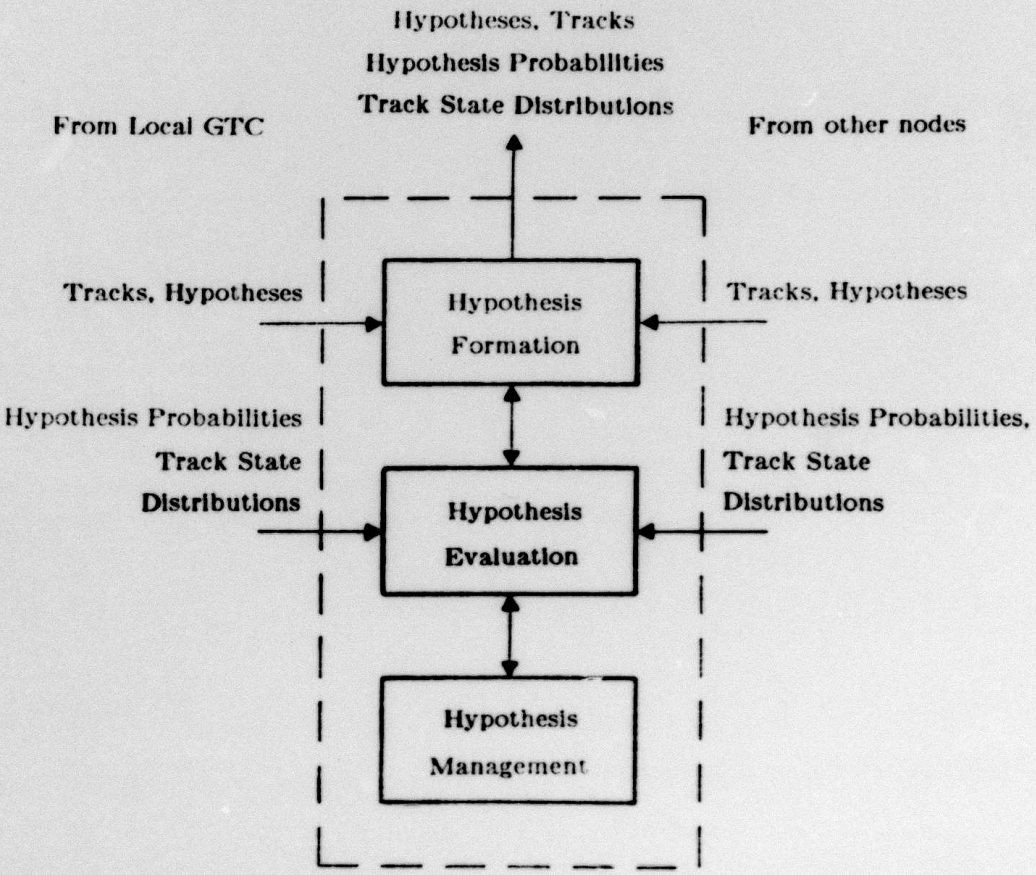


Figure 1-4: Information fusion

system. Information distribution can be fixed a priori for simple systems, or it can be highly adaptive to the information needs in the system.

1.2.4 Resource allocation

This module allocates the resources under the control of the processing node to maintain or improve the performance of the system. Some typical resources include sensor resources and processing resources. Both resource allocation and information distribution can affect the information available in the network. Thus their activities should be coordinated.

1.3 PROJECT GOALS

Many technical issues have to be addressed before DSNs can be designed, built and operated to achieve their military potential. Such issues include the representation and processing of hypotheses, information fusion, communication strategies, resource allocation, adaptation, system architecture, etc. In our previous DSN project, we have successfully addressed some of these issues. The goal of our current effort is to further advance the state of the art in distributed hypothesis testing techniques in DSNs. This will provide more insight as to how a DSN should be designed. Specifically, we intend to accomplish the following technology goals:

1. Develop intelligent distributed algorithms applicable to a wide range of situations such as different network configurations, sensor types, target models, such algorithms should also be adaptive to changing network conditions and make efficient use of sensor resources.

1. Evaluate and adapt these algorithms for real-time implementation.

1. Design experiments to test and evaluate the algorithms in a more realistic scenario such as the Lincoln Laboratory test-bed.

In companion to these technology goals, our plan is to develop a simulation

environment to test the algorithms experimentally on different scenarios

1.4 PROJECT STATUS

There are two parts to our research effort. The first consists of development of algorithms for a DSN and the other is concerned with the development of a simulation environment to test the algorithms and to evaluate the performance of the system experimentally. In the following we discuss their status separately.

We have considered information infusion for DSNs with arbitrary communication patterns among the nodes. The key problems are the formation of possible (or meaningful) global hypotheses from a group of local hypotheses and the evaluation of their probabilities. A set of local hypotheses can be inconsistent so that they cannot be fused to form a global hypothesis. The local probabilities of the local hypotheses may depend on common information which needs to be identified. In the previous project we developed fusion algorithms assuming broadcast communication. In the current project we have obtained fusion algorithms for arbitrary communication. The algorithms are based on modeling the events in the DSN by means of an information graph. To use these algorithms, the histories of the hypotheses and tracks have to be part of the information communicated. Then each node can determine the fusability of the hypotheses and tracks and the common information which has to be removed in evaluating the hypotheses. Information distribution strategies have also been considered. These include strategies which depend only on the local information state and those which model the behavior of other nodes.

The theory of multitarget tracking has been extended to handle targets with a structured state space and dissimilar sensors which observe different components in the target state. The resulting GTC for processing of local sensor data and the information fusion algorithms are very similar to the usual case. However, a multilevel hypothesis formation and evaluation processing architecture is often possible. Consider a network with two nodes. Each node would form hypotheses based on the local measurements and the tracks would be described in the local feature space. During the fusion process, knowledge on the

relationship between the features would be used to generate higher level target tracks from the local feature level tracks. Hypothesis evaluation would then be carried out. As an example, consider the tracking of vehicles. Suppose one sensor node measures only the track/wheel feature and the location. Feature tracks from this node would consist of wheeled or tracked vehicles over time. Suppose another sensor node measures only the location and whether the vehicle has gun or no gun. Tracks generated would consist of gunned or gunless vehicles over time. During fusion, one would use the fact that a vehicle with a gun and track is a tank, a vehicle with neither gun nor track is a truck, etc.

In the previous DSN project, we have concentrated on independent targets. We have now investigated multitarget tracking on structured sets of targets. These include targets which move in groups. One example would be planes flying in formation. Another more complex example consists of military force structures. A division would consist of regiments each of which consists of battalions, and so on. The tracking and identification of such structured targets is important but no much systematic treatment is available. The problem is also interesting in a distributed framework since the observations at different nodes may be at different levels and targets are no longer independent. We have developed models for structured sets of targets, and the notions of multilevel tracks and hypotheses. They are generalizations of our previous work on multitarget tracking which may be viewed as having a single level of targets. Centralized algorithms for evaluating multilevel hypotheses have been obtained. When restricted to two levels with targets moving in independent and identically distributed groups, our results resemble those in single level tracking except the targets in the level are the groups themselves. The main difficulty in implementing these algorithms is in the combinatorics, which becomes more severe with more levels. Thus more practical methods for hypothesis evaluation have to be found. Our future effort would also include finding distributed versions of the algorithms.

The other part of our research effort is concerned with the development of the simulation environment. Since an analytic evaluation of the algorithms and the system performance is difficult our approach is to perform simulation studies. We have developed a simulation system consisting of four DSN nodes with communication patterns which can be specified arbitrarily. Some limited

experimentation on this system has been performed. Our eventual goal for the simulation environment is that it should allow rapid construction of scenarios and rapid development of the DSN system design itself. Also, the environment should be flexible enough to handle various types of processing within each DSN node, including the Bayesian analytic algorithms such as the GTC which have been developed thus far as well as other Artificial Intelligence (AI) based algorithms which would be developed for resource allocation and adaptation. Towards this goal of improving our simulation testbed, we have chosen an AI architecture, called *Schemer* as the basis of the environment. The basic idea behind Schemer is to provide a computational environment in which a system designer can incrementally build his application by specifying the components of the system including the representation of objects that must be known to the system and the procedural rules that define the system's capabilities. To provide this representation support, Schemer is equipped with a general frame representation language for expressing such knowledge. The implementation of this language is currently in progress.

We have investigated the representation of the DSN node within the simulation. Various types of knowledge, such as static knowledge, situational knowledge, planning knowledge, and control knowledge have to be represented. Since the representation of the node in the simulation is a model of the DSN node, we have also obtained a candidate architecture of the node in the actual system. The architecture would be able to carry out both analytic and AI based processing.

1.5 REPORT ORGANIZATION

The rest of this report is organized as follows. In Section 2, we present information fusion algorithms assuming arbitrary communication among the nodes. The algorithms are based on an information graph model of the DSN. Section 3 contains results on tracking of structured targets. Both targets with structured states and structured sets of targets are considered with the emphasis on the latter type. In Section 4 we describe the design and simulation environment. The representation of knowledge in the system and its architecture are discussed. Section 5 presents some simulation results with our current system.

An appendix contains the proofs of the results in Section 2.

2. INFORMATION FUSION FOR ARBITRARY COMMUNICATION

In this section we present algorithms used by each node to fuse the information received from the other nodes with the local information to obtain an updated situation assessment. In [1] fusion algorithms for a broadcast communication pattern were presented. The results of this section extend those algorithms to arbitray communication patterns. In Section 2.1 we describe the information fusion problem in the context of hypothesis formation and evaluation in multitarget tracking. In Section 2.2 a model for information fusion in terms of an information graph is given. Section 2.3 describes the hypothesis formation and evaluation algorithms assuming arbitrary communication.

2.1 THE INFORMATION FUSION PROBLEM

In the following we state the information fusion problem faced by each node in the DSN with emphasis on the relevant issues in multitarget tracking. The formalism is based on the theory of multitarget tracking developed in the previous DSN project [1] and [2].

2.1.1 Local processing

The basic unit of information in the DSN is a *sensor report* $z(t,s)$. This is the output of a sensor s at a time t and is denoted as $((y_j(t,s))_{j=1}^{N_m(t,s)}, N_m(t,s), t, s)$. The index $k=(t,s)$ identifies the sensor report (by time and sensor) uniquely and is called the *sensor report index* or *data index*. $N_m(k)$ is the number of measurements in the report and $(y_j(k))_{j=1}^{N_m(k)}$ is the actual measurement vector. At any given time, let Z be the *data set* consisting of a set of sensor reports and K be the associated *data index set*, i.e, the set of the indices for all the sensor reports contained in Z. The *measurement index set* corresponding to Z is defined as

$$J = \bigcup_{k \in K} \{1,\ldots,N_M(k)\} \times \{k\}. \tag{2.1}$$

Each element $(j,k)=(j,t,s)$ in this set represents the j-th measurement generated at time t by sensor s. The specific value of the measurement is $y_j(t,s)$. According to the system model introduced in Section 1, each node processes the sensor data as they arrive using the *Generalized Tracker/Classifier* (GTC). The output of the GTC when the data is Z consists of the *information state* $\Sigma(Z)$ defined as

$$\Sigma(Z) = (\mathbf{T}(J),(p_t(x \mid \tau,Z))_{\tau \in \mathbf{T}(J)}, \mathbf{H}(J), (P(\Lambda=\lambda \mid Z))_{\lambda \in \mathbf{H}(J)}, \nu(K))$$

where

- $\mathbf{T}(J)$, the set of *possible tracks* defined on J. Each track τ is a subset of J, i.e., $\tau \subseteq J$ and represents the measurement indices coming from a single target. It is usually assumed that a track cannot have two measurement indices in the same sensor report, or the sensor resolution is such that there are no split measurements. Such tracks are then said to be *possible.*

- $p_t(x \mid \tau,Z)$ is the state distribution for a track. Given the track τ, the set of measurements in Z for a hypothesized target is known. From this the distribution of its state x (position, velocity, classification, etc.) at a time t can be found and is a traditional estimation problem. Normally this would be given in terms of a probability distribution; but if the state can be approximated by a Gaussian random vector, the distribution can be expressed in terms of its mean and covariance.

- $\mathbf{H}(J)$ is the set of *possible data-to-data association hypotheses* defined on J. Each *data-to-data association hypothesis* λ is a possible explanation about the origins of all the measurements in Z. Each hypothesis consists of a set of tracks, i.e., $\lambda = \{\tau_1,\tau_2,\ldots\}$. The number of tracks in λ is the number of targets hypothesized to have been detected in the data set Z. Each track τ is the set of measurement indices from a hypothesized target and any measurement index not included in the hypothesis is hypothesized to be a false alarm. We assume that the sensor resolution is such that there are no merged measurements and thus there are no

overlapping tracks in the same hypothesis. The set of hypotheses satisfying this property is said to be *possible*. This represents all *mutually exclusive* and *collectively exhaustive* explanations about the origins of the measurements in Z.

- $P(\Lambda=\lambda \mid Z)$ is the probability of that the true data association Λ is a hypothesis λ given all the measurements in Z. Its computation is the key operation in any multiple hypothesis approach to multitarget tracking and recursive algorithms were given in [1] and [2].

- $\nu(K)$ is the expected number of undetected targets up to and including K. It is important for initiating new tracks. If $\nu(K)$ decreases, the likelihood of any measurement coming from a previously undetected target also decreases.

The information state defined above constitutes a state for multitarget tracking since it contains all the relevant information present in the cumulative data set Z. As long as the information state $\Sigma(Z)$ is known, the GTC can continue to process any new sensor report even though the actual data Z is no longer available. In the GTC, the hypothesis formation module generates the sets $\mathbf{T}(J)$ and $\mathbf{H}(J)$ while the hypothesis evaluation module computes the remaining components in the information state. The hypothesis management module is used to control the combinatorics.

2.1.2 Information Fusion Problem

We assume that each node communicates the information state to the other nodes. Suppose a node receives some messages from the other nodes. It has to fuse or integrate this information with the local information to improve on the local estimate. There are many ways of performing fusion. In our work fusion is based on the following philosophy. The ideal case with the highest performance (but also the highest communication cost) is when the nodes communicate the actual sensor data through the network instead of the processes information. In this case a node would be able to generate an optimal information state based on

all the data available. Since in a more realistic DSN only the information states are communicated, an appropriate objective for fusion is to reconstruct the optimal information state based on the information states received from the other nodes. To facilitate further discussion, we call the data available to each node before communication takes place as local data and the maximum data set available after communication as global data. Local and global information states, hypotheses, tracks, etc. are all defined analogously.

There are thus two steps to the fusion process. The first step in the fusion process consists of generating the possible track and hypothesis sets based on the global data from the local tracks and hypotheses. Since the local data are the part of the global data available to the nodes at the given times, the global tracks and hypotheses when restricted to the local data should give the local tracks and hypotheses. This implies that a certain combination of local tracks and hypotheses should not be fused, i.e., there may not exist global tracks and hypotheses for given sets of local tracks and hypotheses. In Figure 2-1, the two tracks τ_1 and τ_2 are two local tracks maintained at two different nodes. They cannot be fused since the resulting global track would have two different measurements in the same sensor report 1, thus violating the no split measurement assumption. On the other hand, τ_1 and τ_3 can be fused to yield a global track $\tau_1 \bigcup \tau_2$. The interpretation of this global track is that the measurements in both tracks τ_1 and τ_3 come from the same target. Tracks τ_1 and τ_4 can also be fused. However, they do not have to be and in that case the two tracks correspond to two different targets. The fusability question also needs to be addressed at the hypothesis level. Each local hypothesis is a possible explanation about the origins of the local measurements. Thus if the local hypotheses are incompatible, they cannot be fused to obtain a global hypothesis. This is illustrated in Figure 2-2 where each node i has two local hypotheses λ_i^j, $j=1,2$ derived from the two common hypotheses λ^j, $j=1,2$. Since λ^1 and λ^2 are mutually exclusive, the local hypotheses λ_1^2 and λ_2^1 are not fusable.

The second step in the fusion process consists in generating the state distributions of the global tracks and the probabilities of the global hypotheses using the local distributions and probabilities. If the nodes communicated in the past, the local statistics would not be independent. A key problem in fusion is to identify the common information shared by the nodes and make sure it is not used

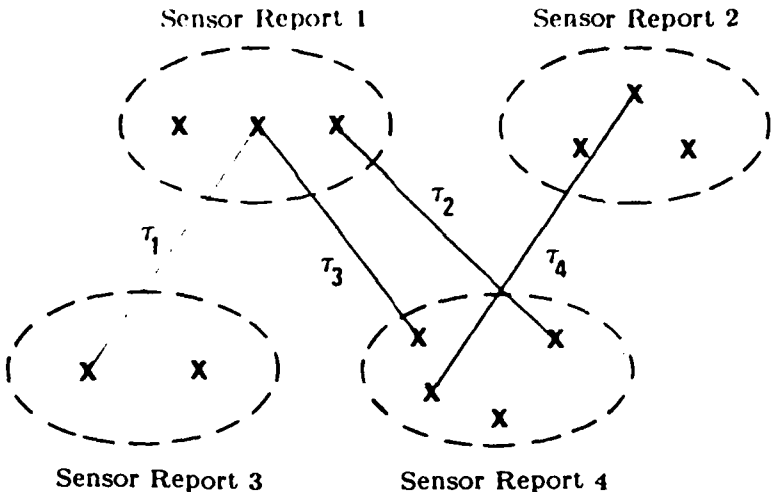


Figure 2-1: Fusability of tracks

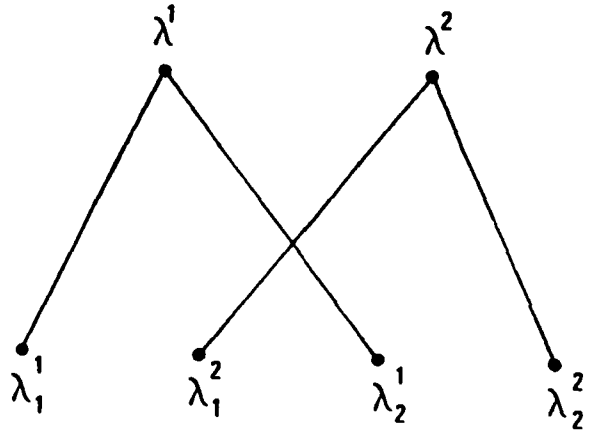


Figure 2-2: Fusability of hypotheses

more than once in generating the global statistics.

2.2 INFORMATION GRAPH

In performing information fusion, it is necessary to identify the information available to the nodes in the network at various times and how the information of one node at one time is related to that of another node at a different time. For example, whenever two nodes communicate some common information is shared between the nodes. The existence of this shared information would have to be recognized in any future information fusion. Specifically, before any global hypothesis can be generated, the fusability of the local hypotheses have to be checked based on their histories. Furthermore, when the probabilities of the hypotheses are to be evaluated, the common information should only be used once. This necessitates tracking the histories of the communication and can be accomplished conveniently using the information graph. The information graph introduced below can also be viewed as an abstract model for a DSN.

2.2.1 Information graph model

We assume that there is a set of processing nodes called N. Each node n in N receives data from a set of sensors called S_n such that $S_n \cap S_{n'}=\emptyset$ for $n \neq n'$, i.e., each sensor s only reports to one processing node. Let $\mathbf{S}=\bigcup_{n\in\mathbf{N}} S_n$ be the set of all sensors. If a sensor s generates a report at time t with value z, the report is denoted as (z,t,s) or simply $z(t,s)$. Each sensor report is the basic unit of information and the set of all such reports is denoted by Z called the *total information or data set.* Each sensor report is indexed by $k=(t,s)$, i.e., the time t when it is generated and the sensor s responsible for its generation. The set of all such indices is called the *total data index set* and denoted as

$$\mathbf{K}=\{(t,s) \mid (z,t,s)\in\mathbf{Z} \text{ for } \textit{some } z\} \tag{2.2}$$

At any one time, a node's information may consist of only a subset Z of Z. Such a Z is called a *partial information set* or *partial data set*, or simply *information set* or *data set.* For each Z there is a K correponding to the data indices in Z.

The sensors send the data intantaneously to the nodes as soon as they are generated. The communication among the nodes can be characterized by the *communication schedule* C which is a subset of $T \times T \times N \times N$. An element (t, t', n, n') means that the communication transmitted at time t by node n is received at time t' by node n'.

The information at each sensor or node in the DSN is affected by four types of events. The nature of the events, the times at which they occur and the nodes affected are given below:

1. Sensor observation and transmission – $I_{ST} = K \times \{ST\}$

2. Sensor data received at node -- $I_{SR} = K \times \{SR\}$

3. Transmission of communication by node --
   $I_{CT} = \{(n, t, CT) \mid (t, t', n, n') \in C\}$

4. Reception of communication by node --
   $I_{CR} = \{(n, t, CR) \mid (t', t, n', n) \in C\}$

Let I be defined as

$$I = I_{ST} \cup I_{SR} \cup I_{CT} \cup I_{CR} \tag{2.3}$$

I constitutes all the significant events in the network and forms the set of information nodes (not DSN nodes) in the information graph. To represent the relation between these nodes, we define an antisymmetric, reflexive and transitive binary relation $\leq$ on I as follows: for any i and i' in I, $i \leq i'$ if $i = i'$ or there is a communication path from i to i'. The *information graph* on the system is then the ordered set $(I, \leq)$. By using the graph we can determine how the information in the system flows. In particular, it is easy to find the history of the information at a certain node. As we shall see later, this is useful for the purpose of information fusion.

Figure 2-3 show the information graph for broadcast communication. At a given time all the nodes communicate to each another so that they all have the same information after that. Figure 2-4 shows the information graph for a cyclic

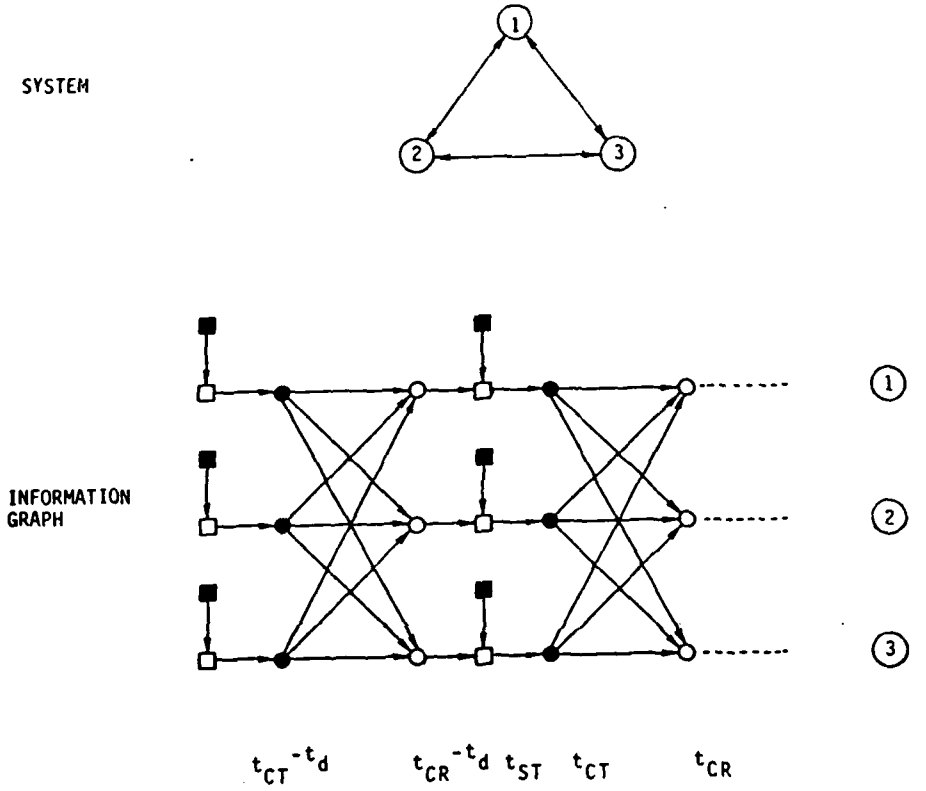


Figure 2-3: Broadcast communication

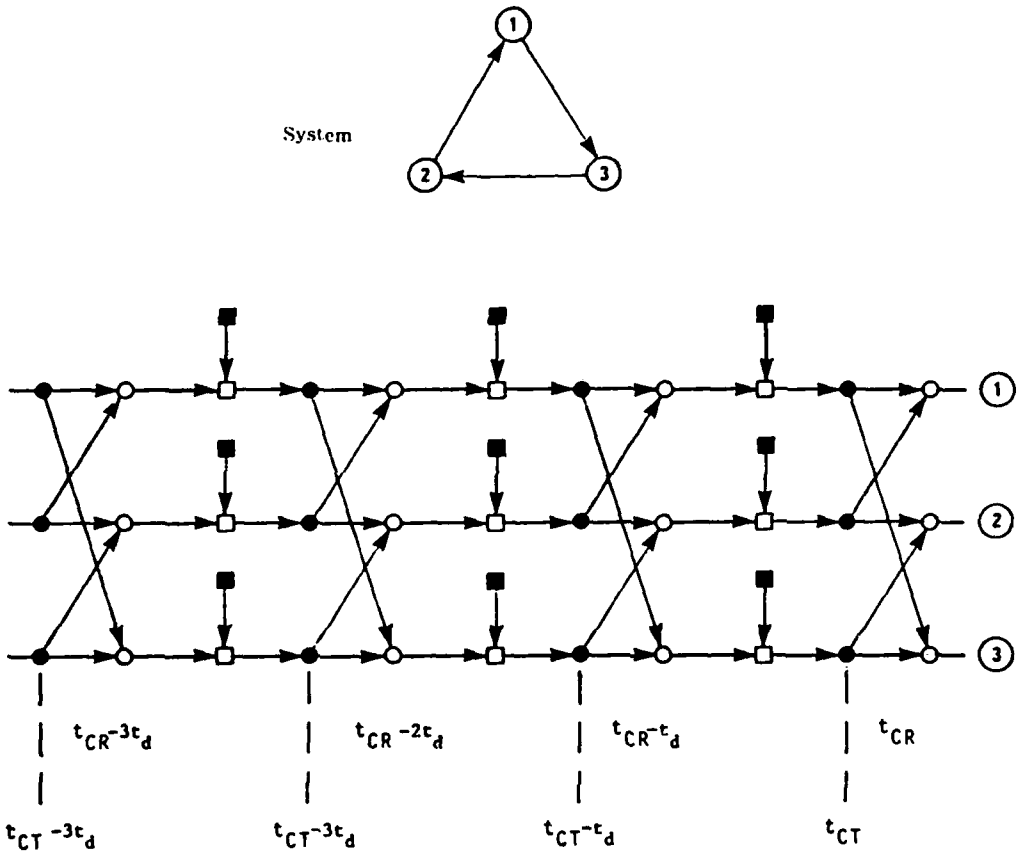


Figure 2-4: Cyclic communication

communication system. The system consists of three nodes $\mathbf{N}=\{1,2,3\}$ collecting data from the three sensors $\mathbf{S}=\{1,2,3\}$, respectively at the times $\ldots,t_{ST},t_{ST}+t_d,\ldots\ldots$ The nodes transmit to the other nodes periodically according to the pattern shown in Figure 2-4 at times $\ldots,t_{CT},t_{CT}+t_d,\ldots$ and the messages are received at the times $\ldots.t_{CR},t_{CR}+t_d,\ldots.$ It is assumed that $t_{ST}<t_{CT}<t_{CR}$.

For each information node i in the information graph, the maximum amount of information available is the sensor data that would be received if they had been communicated in the network. Thus associated with each node i the *(maximum) data index set* K_i and the *(maximum) information set* Z_i are defined as follows:

$$K_i = \{k \in \mathbf{K} \mid (k,ST) \leq i\} \tag{2.4}$$

$$Z_i = \{(z,k) \in \mathbf{Z} \mid k \in K_i\}. \tag{2.5}$$

As stated before, our philosophy is to assume that each node tries to reconstruct the best estimate as if all sensor data are transmitted. Thus from now on the information available at each node i is assumed to be Z_i with the data index set K_i.

The following observations are quite obvious from the definitions:

1. $K_i = \{k \in \mathbf{K} \mid (k,SR) \leq i\}$ for all i in **I**.

2. $K_i \subseteq K_{i'}$ if $i \leq i'$. (The information of a node always includes that of any predecessor node.)

3. $K_i = \bigcup_{i' \leq i} K_{i'}$ for all i in **I**. (The information at a node is the union of that of the predecessors.)

4. $K_i = \bigcup_{i' \rightarrowtail i} K_{i'}$ for all i in **I**, where $i' \rightarrowtail i$ means that i' is the immediate predecessor of i. (One needs only to consider the immediate predecessors of i in generating the information available to i.)

Since there is a one-to-one correspondence between K and Z, a similar set of observations can be made for Z.

1. $Z_i = \{Z \in \mathbf{Z} \mid (Z, SR) \leq i\}$ for all i in I.

2. $Z_i \subseteq Z_{i'}$ if $i \leq i'$.

3. $Z_i = \bigcup_{i' \leq i} Z_{i'}$ for all i in I.

4. $Z_i = \bigcup_{i' \rightarrow i} Z_{i'}$ for all i in I.

Consider an information node $i_0 \in \mathbf{I_{CR}}$. This represents the event that communication from other nodes is received. Let I be the set of immediate predecessor nodes for i_0. The fusion problem is to find the information state of i_0 using the information states of the nodes in I (and those of other predecessor nodes of I, if necessary). As mentioned before, it is important to identify the common information in the data represented by I. This turns out to be

$$\bigcap_{i \in I} K_i = \bigcup_{i' \in C(I)} K_{i'} \tag{2.6}$$

where

$$C(I) = \{i' \in \mathbf{I} \mid i' \leq i \text{ for } \textit{all } i \in I\} \tag{2.7}$$

is the set of common predecessors for all the nodes in I. Equation (2.6) states that the common information shared by the nodes in I is the union of the information of the common predecessor nodes of I. In fact, based on the observation (4) above, $C(I)$ can be replaced by $C_{\max}(I)$ which is the maximum set in $C(I)$ with respect to the set-inclusion partial order whereby $I_1 < I_2$ when $I_1 \subset I_2$ and $i_1 < i_2$ for all $i_1 \in I_1$ and $i_2 \in I_2$. Then the union needs to be taken only over the set $C_{\max}(I)$, i.e., equation (2.6) becomes

$$\bigcap_{i \in I} K_i = \bigcup_{i' \in C_{\max}(I)} K_{i'} \tag{2.8}$$

If necessary, we can regard $C_{\max}(I)$ as I in equation (2.8) and repeat the process to find the common information shared by all the nodes in $C_{\max}(I)$. This would be used in the following section to develop distributed estimation algorithms.

2.2.2 Distributed estimation

We now consider the distributed estimation problem to illustrate the use of the information graph. Any uncertainty in the origins of the measurements is ignored for the time being. The results would be useful in the next subsection when we consider distributed multitarget tracking.

The state to be estimated is a random vector x. The *a priori* probability density (or distribution) is $p(x)$. The observation generated by a sensor s at time t is $z(t,s)$. The following additional assumptions are needed:

- Both the sensor schedule **K** and the communication schedule **C** are independent of the state x.

- Given x and **K**, each element in **Z** is conditionally independent from each other and has an absolutely continuous transitional probability from state x to measurement.

The distributed estimation problem is then to compute $p(x \mid Z_i)$ for each $i \in \mathbf{I}$. From the definition of I, this needs only to be carried out for the sets $\mathbf{I}_{SR}$ and $\mathbf{I}_{CR}$ since the only activities at the other nodes involve transmission. For an information node in $\mathbf{I}_{SR}$, we have a traditional Bayesian update problem where the conditional probability is updated using the sensor report. We are primarily interested in a problem involving information nodes in $\mathbf{I}_{CR}$. Suppose the information node of interest is i_0 and that the immediate predecessors of i_0 form the set I. Then

$$Z_{i_0} = \bigcup_{i \in I} Z_i \tag{2.9}$$

The objective is the computation of $p(x \mid \bigcup_{i \in I} Z_i)$ in terms of the predecessor probabilities $p(x \mid Z_{i'})_{i' \leq i_0}$. Ideally, one would like to use only the probabilities defined on I, but as we shall see, this is not always possible.

In the appendix, we show that

$$p(x \mid \bigcup_{i=1}^{n} Z_i) = c \prod_{i=1}^{n} \Big(\prod_{N \in \mathbf{N}_i^n} p(x \mid \bigcap_{j \in N} Z_j) \Big)^{(\cdot)^{(\cdot)}} \tag{2.10}$$

where c is a normalization constant and

$$\mathbf{N}_i^n = \{N \subseteq \{1,...,n\} \mid \#(N)=i\} \tag{2.11}$$

is the set of all subsets of $\{1,...,n\}$ with i elements. In equation (2.11), $\#(N)$ denotes the number of elements in the set N. For $n=2$, this yields the fusion formula for two nodes:

$$p(x \mid Z_1 \cup Z_2) = c\frac{p(x \mid Z_1)p(x \mid Z_2)}{p(x \mid Z_1 \cap Z_2)} \tag{2.12}$$

Equation (2.11) can be interpreted as follows. Since the probabilities $p(x \mid Z_1)$ and $p(x \mid Z_2)$ both utilize the information contained in $Z_1 \cap Z_2$, the division by $p(x \mid Z_1 \cap Z_2)$ is needed to remove the common information so that it is used only once. Equation (2.10) is just a general form where the probabilities from multiple nodes are to be fused. Unfortunately, in both (2.10) and (2.12) there are still terms involving intersections of the Z_i's. If all these intersections are of the form Z_j for some information node j or empty corresponding to the common a priori information, then equation (2.10) or (2.12) serves as a fusion algorithm. In this algorithm, the conditional probability at the fusion node is a product and ratio of the conditional probabilities defined on a set of predecessor nodes. From the definition of the information graph, all these probabilities can be communicated.

If there is an intersection $\bigcap_{j\in N} Z_j$ which is not equal to $Z_{j'}$ for some $j' \in \mathbf{I}$, then by (2.6) the intersection can be expressed as the union of the information of some information nodes again. Equation (2.10) can then be applied to evaluate the probability $p(x \mid \bigcap_{j\in N} Z_j)$. The process can be repeated until all the probabilities are either conditioned on the information at the individual information nodes or the a priori information. For notational convenience, we represent the a priori information by adding an element $i_\emptyset$ to the set $\mathbf{I}$ of all the information nodes and let $\bar{\mathbf{I}}=\mathbf{I}\cup\{i_\emptyset\}$. Then the extended information graph $(\bar{\mathbf{I}},\leq)$ is constructed by letting $i_\emptyset$ be the immediate predecessor of all the minimum nodes in the original information graph $(\mathbf{I},\leq)$. Then we have $Z_{i_\emptyset}=K_{i_\emptyset}=\emptyset$. With this definition it can be shown in the appendix that

$$p(x \mid \bigcup_{i\in I} Z_i) = C \prod_{\bar{i}\in\bar{I}} p(x \mid Z_{\bar{i}})^{\alpha(\bar{i})} \tag{2.13}$$

where $\bar{I}\leq I$ is a subset of $\bar{\mathbf{I}}$, $(\alpha(\bar{i}))_{\bar{i}\in\bar{I}}$ is some index tuple such that $\alpha(\bar{i})$ is a

nonzero integer for each $\bar{i}$, and C is the normalizing constant. The set $\bar{I}$ contains all the information nodes which are relevant to fusion at node i_0. $\alpha(\bar{i})$ decides whether the information at node $\bar{i}$ should be added ($\alpha(\bar{i})=1$) or removed ($\alpha(\bar{i})=-1$).

To illustrate the use of this algorithm, let us first consider a broadcast communication pattern of Figure 2-3. For notational simplicity, we would suppress the type of the node in naming the node. Consider the information node (t_{CR},n). We have

$$\bigcap_{n\in N} Z(t_{CT},n)=Z(t_{CR}-t_d,n). \tag{2.14}$$

Thus, the fusion algorithm for a node n at time t_{CR} is

$$p(x \mid Z(t_{CR},n)) = C\prod_{i\in N}\frac{p(x \mid Z(t_{CT},i))}{p(x \mid Z(t_{CR}-t_d,i))}p(x \mid Z(t_{CR}-t_d,n)) \tag{2.15}$$

where C is a normalizing constant. Each term in the product is the new information contained in the sensor report $z(t_{ST},i)$.

For the cyclic communication system shown in Figure 2-4, consider node 1 at time t_{CR}. The immediate predecessors of the information node $(t_{CR},1)$ are $(t_{CT},1)$ and $(t_{CT},2)$. Equation (2.12) can thus be used to find $p(x \mid Z(t_{CR},1))$. From the information graph of Figure 2-4, the common predecessors of $(t_{CT},1)$ and $(t_{CT},2)$ consist of the two nodes $(t_{CT}-2t_d,1)$ and $(t_{CT}-t_d,2)$. Thus

$$Z(t_{CT},1)\bigcap Z(t_{CT},2) = Z(t_{CT}-2t_d,1)\bigcup Z(t_{CT}-t_d,2), \tag{2.16}$$

and equation (2.12) can be used to find the probability of the right hand side again. From the information graph,

$$\begin{aligned} Z(t_{CT}-2t_d,1)\bigcap Z(t_{CT}-t_d,2) &= Z(t_{CT}-3t_d,1)\bigcup Z(t_{CT}-3t_d,2) \\ &= Z(t_{CR}-3t_d,1). \end{aligned} \tag{2.17}$$

Thus, the algorithm gives for general $i=1,2,3$

$$\begin{aligned} p(x \mid Z(t_{CR},i)) = C\frac{p(x \mid Z(t_{CT},i))}{p(x \mid Z(t_{CT}-2t_d,i))}\frac{p(x \mid Z(t_{CT},|i+1|))}{p(x \mid Z(t_{CT}-t_d,|i+1|))} \\ \times p(x \mid Z(t_{CR}-3t_d,i)) \end{aligned} \tag{2.18}$$

where $[i]$ is i modulo 3.

This is in the form of equation (2.13) with five nodes in the set $\bar{I}$. Thus, in addition to its current conditional probability $p(x \mid Z(t_{CT},1))$, and $p(x \mid Z(t_{CT},2))$ which comes from node 2, node 1 has to store three other probabilities. Note that $p(x \mid Z(t_{CT}-t_d,2))$ is available to node 1 from earlier communications. This indicates that in a distributed sensor network, knowing the most recent estimate may not be sufficient if one wants to recover the globally optimal estimate.

Our discussion above assumes the fusion algorithm for each node is provided by a system designer based on the information graph. Alternatively, we may assume that the information graph is known to all the DSN processing nodes who then compute the algorithms in a distributed manner. Still another possibility is for each message to contain a history of the nodes and times that it has passed through. Then a fusion node can use the histories of the messages received to construct a partial information graph so that fusion can be performed. This philosophy would be useful for fusion when the communication pattern is not fixed a priori, such as when nodes can vary their communiation strategies or have to adapt to system failures.

2.3 FUSION IN MULTITARGET TRACKING

In this section we consider the fusion algorithm for multitarget tracking assuming arbitary communication pattern. The algorithm is based on the theory of multitarget tracking developed under the previous project [1] and the concept of the information graph. In the previous project [1], the information fusion in multitarget tracking was investigated primarily for broadcast type communication pattern. In this section, we treat the same subject assuming an arbitrary communication pattern which is defined in terms of an information graph.

2.3.1 Problem formulation

In Section 2.1 we introduced the fusion problem in general terms. We now state it more formally in terms of an information graph. Given the communication pattern of the network, an information graph is defined. For each information node i in the graph, there is a data index set K_i and an information set or data set Z_i as defined before. Since we are now interested in multitarget tracking, we have to deal with measurement index sets on which tracks and hypotheses are defined. A measurement index set J_i at an information node i is defined as

$$J_i = \{(j,k) \in \mathbf{J} \mid k \in K_i\}.$$

The activities in a DSN can be represented by the expansion of the nodes in the information graph. Two types of nodes, namely those in $\mathbf{I}_{ST}$ and $\mathbf{I}_{CT}$, involve only communication. For the other two types, namely the ones in $\mathbf{I}_{SR}$ and $\mathbf{I}_{CR}$, information processing is involved. At a node in $\mathbf{I}_{SR}$, the data received from the local sensors are processed by each node using the GTC, producing an information state for the node. For a node $i_0 \in \mathbf{I}_{CR}$, messages are received from other nodes in the DSN and fusion takes place. *Let I be the set of immediate predecessor nodes of i_0.* For any node i in I, assume the possible tracks $\mathbf{T}(J_i)$ and the possible hypotheses $\mathbf{H}(J_i)$ are known. In addition to these, the local probabilities of the tracks and hypotheses are also given. From the information graph, the measurement index set for the information node i_0 is $J = \bigcup_{i \in I} J_i$. The two specific subproblems in information fusion are then the following:

- (Hypothesis formation) How should node i_0 construct the *possible* (global) track set $\mathbf{T}(J)$ and the *possible* (global) hypothesis set $\mathbf{H}(J)$?

- (Hypothesis evaluation) Suppose the global sets of tracks and hypotheses are formed. How can we evaluate the probability of each hypothesis using the probabilities of the predecessor nodes? Also, how should the state distributions of the tracks be computed?

The two problems would now be discussed separately.

2.3.2 Hypothesis formation

As we discussed before in Section 2.1, not all local tracks and hypotheses can be fused to form meaningful global tracks and hypotheses. Our philosophy behind information fusion is to reconstruct the information state $\Sigma(Z)$ starting from the information states $\Sigma(Z_i)$. This means that two tracks can only be fused if there exists a global track which is consistent with them. This is also the idea behind the fusion of hypotheses. The following are some definitions needed to formalize this concept.

Consider any two measurement index sets J_1 and J_2 with $J_2 \subseteq J_1$. For each track τ in $\mathbf{T}(J_1)$ the *restriction* of the track τ on J_2 is defined as $\tau \cap J_2$, i.e., the track consisting of only those measurement indices in J_2. We usually say that the track τ is a *successor* of its restriction $\tau \cap J_2$ or conversely, $\tau \cap J_2$ is the *predecessor* track of τ. Similarly, for each hypothesis λ in $\mathbf{H}(J_1)$, the *restriction* of the hypothesis λ on J_2 is defined to be

$$\lambda \mid J = \{\tau \cap J \mid \tau \in \lambda\} \setminus \{\emptyset\} \tag{2.19}$$

i.e., a *hypothesis whose tracks* are those of λ restricted to J_2. The concepts of predecessor and successor hypotheses can be defined as in tracks.

Let $(J_i)_{i \in I}$ be an arbitrary tuple of measurement index sets where I is an arbitrary nonempty set. I does not have to be related to the information graph at all. Then any tuple $(\tau_i)_{i \in I}$ of tracks in $\prod_{i \in I} \mathbf{T}(J_i)$ is said to be *fusable* if there exists a track τ in $\mathbf{T}(\bigcup_{i \in I} J_i)$ such that

$$\tau \cap J_i = \tau_i \tag{2.20}$$

for all $i \in I$. τ is a track obtained by fusing the tracks in the tuple. Similarly any tuple $(\lambda_i)_{i \in I}$ of hypotheses in $\prod_{i \in I} \mathbf{H}(J_i)$ is said to be *fusable* if there exists a hypothesis λ in $\mathbf{H}(\bigcup_{i \in I} J_i)$ such that

$$\lambda \mid J_i = \lambda_i \tag{2.21}$$

for all $i \in I$. Fusability of tracks thus means that there exists a possible global track such that each of the local tracks represents a restriction of the global track to the local measurement indices. Similarly the fusability of the hypotheses

means there exists a global hypothesis such that each local hypothesis is a restriction of the global hypothesis to the local measurement index set, or more specifically, the nonempty restrictions of the tracks in the global hypothesis are the local hypotheses.

If the measurement index sets $(J_i)_{i\in I}$ do not intersect, fusability of tracks and hypotheses is trivially assured. When the measurement index sets do overlap, we have to be concerned about the consistency in the tracks and hypotheses. The following rather intuitive conditions for checking fusability are proved in the appendix.

1. Any track tuple $(\tau_i)_{i\in I}$ in $\prod_{i\in I}\mathbf{T}(J_i)$ is fusable if and only if

$$\tau_{i_1}\bigcap(J_{i_1}\bigcap J_{i_2}) = \tau_{i_2}\bigcap(J_{i_1}\bigcap J_{i_2}) \tag{2.22}$$

for all $(i_1,i_2)\in I\times I$.

2. Any hypothesis tuple $(\lambda_i)_{i\in I}$ in $\prod_{i\in I}\mathbf{H}(J_i)$ is fusable if and only if

$$\lambda_{i_1}\,|\,(J_{i_1}\bigcap J_{i_2}) = \lambda_{i_2}\,|\,(J_{i_1}\bigcap J_{i_2}) \tag{2.23}$$

for all $(i_1,i_2)\in I\times I$.

These two conditions state that a tuple of tracks (or hypotheses) is fusable if and only if they share common predecessors (in tracks or hypotheses) in the overlapping measurement index set

$$\bar{J} = \bigcup\{J_{i_1}\bigcap J_{i_2}\,|\,(i_1,i_2)\in I\times I \text{ such that } i_1\neq i_2\} \tag{2.24}$$

To check the conditions described by (2.23) or (2.24), we need to have tracks and hypotheses defined on the set $\bar{J}$. In general, these are not directly available since there may not be any information node with $\bar{J}$ as its measurement index set. However, by using the decomposition algorithm of equation (2.6), we can express the set $\bar{J}$ as the union of the measurement index sets of some predecessor nodes in the information graph. The two fusability conditions of equations (2.22) and (2.23) can be further reduced to the following.

Let i_0 be a communication receiving node and I be the set of all the immediate predecessors of it. For each $(i_1,i_2)\in I\times I$, let $\bar{I}(i_1,i_2)$ be a set of information nodes $\bar{i}$ such that $\bar{i}<i_1$ and $\bar{i}<i_2$, i.e., their common predecessor nodes. Then, we have

1. a necessary and sufficient condition for any track tuple $(\tau_i)_{i\in I}\in\prod_{i\in I}\mathbf{T}(J_i)$ to be fusable is that, for any $(i_1,i_2)\in I\times I$,

$$\tau_{i_1}\bigcap J_{(\bar{i})}=\tau_{i_2}\bigcap J_{(\bar{i})} \tag{2.25}$$

for any $\bar{i}\in\bar{I}(i_1,i_2)$, and

2. a necessary condition for any hypothesis tuple $(\lambda_i)_{i\in I}\in\prod_{i\in I}\mathbf{H}(J_i)$ to be fusable is that, for any $(i_1,i_2)\in I\times I$,

$$\lambda_{i_1}\mid J_{(\bar{i})}=\lambda_{i_2}\mid J_{(\bar{i})} \tag{2.26}$$

for any $\bar{i}\in\bar{I}(i_1,i_2)$.

In general, for any two distinct nodes i_1 and i_2, their common predecessor set $\bar{I}(i_1,i_2)$ may not be unique. However, to use the above conditions to test the fusability, we need only to consider the set of all the maximum elements in the set $\{\bar{i}\in\mathbf{I}\mid\bar{i}<i_1\text{and}\bar{i}<i_2\}$, i.e., the maximum common predecessor set. Thus in the cyclic communication example of Figure 2-4, a track from the node $(t_{CR},1)$ and one from the node $(t_{CR},2)$ are fusable if and only if they have the same predecessor (or restriction) tracks in both the nodes $(t_{CT}\ 2,1)$ and $(t_{CT}\ 1,2)$.

The test defined by (2.25) provides a necessary and sufficient condition for track fusability but equation (2.26) only provides a necessary condition for hypothesis fusability. This is due to the fact that a fusable tuple of tracks produces only one fused track but a fusable tuple of hypotheses may produce more than one hypotheses. The counterexample in Figure 2-5 shows that (2.26) is not a sufficient condition for the hypothesis fusability. In this example, the two hypotheses (λ_1,λ_2) are to be fused. The common predecessors of the nodes 1 and 2 are nodes 3 and 4. It is obvious that $\lambda_1\mid J_3=\lambda_2\mid J_3$ and also $\lambda_1\mid J_4=\lambda_2\mid J_4$, thus satisfying the necessary condition of (2.26) for hypothesis fusability. In fact,

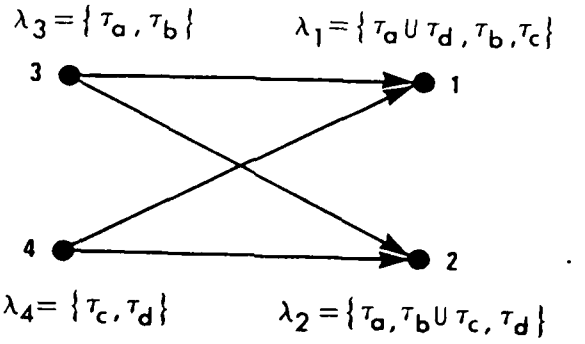


Figure 2-5: Counterexample

this is true since both λ_1 and λ_2 are the results of fusing λ_3 and λ_4. However, since

$$\lambda_1 \mid J_3 \cap J_4 \neq \lambda_2 \mid J_3 \cap J_4 \tag{2.27}$$

the hypothesis fusability condition of (2.23) is violated. This is again obvious since λ_1 and λ_2 are mutually exclusive. λ_1 hypothesizes that τ_a and τ_d are from the same target whereas λ_2 hypothesizes that τ_a and τ_d are from different targets.

Although it is not sufficient to determine hypothesis fusability by considering only the predecessors of the hypotheses in the predecessor nodes, the condition (2.26) can be used to eliminate hypotheses for further consideration if they do not have the same predecessor hypothesis in a common predecessor node. Furthermore, the following equivalence condition, proved in the appendix, relates hypothesis fusability to track fusability.

Hypothesis Fusability Condition. Let $(J_i)_{i\in I}$ be any tuple of measurement index sets and $J=\bigcup_{i\in I} J_i$. Then, any $(\lambda_i)_{i\in I}\in\prod_{i\in I}\mathbf{H}(J_i)$ is fusable with fused hypothesis $\lambda\in\mathbf{H}(J)$ if and only if

1. for any τ in λ, there exists a fusable track tuple $(\tau_i)_{i\in I}\in\prod_{i\in I}(\lambda_i\cap\{\emptyset\})$ such that $\tau=\bigcup_{i\in I}\tau_i$, and

2. for all $i\in I$ and for all $\tau_i\in\lambda_i$, there exists a unique τ in λ such that $\tau_i\subseteq\tau$.

Condition 1 states that every track τ in the hypothesis λ is formed by taking the union of the fusable tracks in the local hypotheses. Condition 2 states that every τ_i belongs to a unique global track in any given global hypothesis.

Hypothesis formation thus consists of the following steps:

1. Use the necessary condition of (2.26) to reduce the candidates for fusable hypothesis tuples

2. Use the track fusability condition of (2.25) to further determine hypothesis fusability

3. Exhaust all possible fusable hypothesis tuples, and for each fusable hypothesis tuple, generate all possible fused hypotheses.

The last step is concerned with the actual hypothesis formation and consists of a two-level procedure. The first level performs hypothesis-to-hypothesis association. The second level carries out the actual track-to-track association to form global tracks from the fusable track tuples.

2.3.3 Hypothesis evaluation

Given the global hypotheses and global tracks constructed from the local hypotheses and local tracks, the objective of hypothesis evaluation is to compute their probabilities and state distributions using the communicated local information. In terms of the information graph, the problem is as follows. Let $i_0 = (t, n, CR)$ be a communication receiving node in $\mathbf{I}_{CR}$ and I be the set of all the immediate predecessors of i_0. Let $Z = \bigcup_{i \in I} Z_i$ with K and J be the associated index set and measurement index set. We need to compute the probabilites of all hypotheses. $(P(\Lambda = \lambda \mid Z))_{\lambda \in \mathbf{H}(J)}$, the state distributions of the tracks, $(p_t(x \mid Z, \tau))_{\tau \in \mathbf{T}(J)}$, and the expected number $\nu(K)$ of undetected targets.

We make the standard assumptions on the target and sensor models (see [1] or [2]). In particular, the target models are assumed to the independent and identically distributed Markov processes and the number of targets is Poisson distributed. The sensor measurements generated by sensors at different times are conditionally independent given the target state. In addition to these, we also make the special assumption that the target state is either static or bidirectionally deterministic (which makes it equivalent to a static process). This assumption is needed to make the algorithm more implementable. Later in this section, we would briefly discuss how this assumption can be relaxed. The target state is in a hybrid variable with a continuous part to model geolocation type variables and a discrete part to model classification type information. For convenience, we define a hybrid measure μ on the state space to be the direct product of a continuous measure and a discrete measure. Then any integral with respect to this hybrid measure is a sum of integrals over the continuous part of the state space.

With these assumptions, the following hypothesis evaluation results are derived in the appendix. Let $(\bar{I}, \alpha)$ be the pair which satisfies the condition (2.13) of Section 2.2.2. Suppose for each $\bar{i} \in \bar{I}$, the probability $p(\lambda \mid Z_{\bar{i}})$ for each hypothesis λ in $\mathbf{H}(J_{\bar{i}})$, the track state distribution $p(x \mid Z_{\bar{i}}, \tau)$ for each track τ in $\mathbf{T}(J_{\bar{i}})$, and $\nu(K_{\bar{i}})$, the expected number of undetected targets are all known. Then for every hypothesis $\lambda \in \mathbf{H}(J)$, the probability of the hypothesis being true is given by

$$P((\Lambda \mid J)=\lambda \mid Z) = C^{-1} \prod_{\bar{i}\in\bar{I}} P((\lambda \mid J_{\bar{i}}) \mid Z_{\bar{i}}^{\alpha(\bar{i})} \prod_{\tau\in(\lambda \mid J)} \tilde{L}(\tau,(Z_{\bar{i}})_{\bar{i}\in\bar{I}}) \tag{2.28}$$

where C is a normalization constant, and

$$\tilde{L}(\tau,(Z_{\bar{i}})_{\bar{i}\in\bar{I}}) = \int \prod_{\bar{i}\in\bar{I}} \tilde{p}(x \mid Z_{\bar{i}},(\tau\cap J_{\bar{i}}))^{\alpha(\bar{i})}\mu(dx) \tag{2.29}$$

is the likelihood of the global track τ. The expected number of undetected targets is given by

$$\nu(K) = \tilde{L}(\emptyset,(Z_{\bar{i}})_{\bar{i}\in\bar{I}}) = \int \prod_{\bar{i}\in\bar{I}} \tilde{p}(x \mid Z_{\bar{i}},\emptyset)^{\alpha(\bar{i})}\mu(dx) \tag{2.30}$$

where

$$\tilde{p}(x \mid Z_{\bar{i}},\tau)=p(x \mid Z_{\bar{i}},\tau)\nu(K_{\bar{i}})^{\epsilon_{\bar{i}}(\tau)}, \tag{2.31}$$

$$\epsilon_{\bar{i}}(\tau) = \begin{cases} 1 & \text{if } \tau\cap J_{\bar{i}}=\emptyset \\ 0 & \textit{otherwise} \end{cases} . \tag{2.32}$$

The state distribution of the track τ can be updated by

$$p(x \mid \tau,Z) = c^{-1} \prod_{\bar{i}\in\bar{I}} p(x \mid (\tau\cap J_{\bar{i}}),Z_{\bar{i}})^{\alpha(\bar{i})} \tag{2.33}$$

where c is a normalization constant.

We note first of all that hypothesis evaluation depends only on the statistics at the information nodes in the set $\bar{I}$. This is the same set used in distributed estimation and represents the nodes which are relevant for fusion. The function α determines whether the information at a node should be added or substracted. The hypothesis evaluation formula of (2.28) has a two-level structure. At the higher level, the product of the local hypothesis probabilities evaluates the probability of associating the given set of local hypotheses. The next level consists of the likelihoods of the individual tracks.

Each $\tilde{L}(\tau,(Z_{\bar{i}})_{\bar{i}\in\bar{I}})$ is a track-to-track association likelihood, i.e., the likelihood of associating all the tracks in the local track tuple $(\tau\cap J_{\bar{i}})_{\bar{i}\in\bar{I}}$ with one target represented by the global track τ which is their union. Its evaluation depends on the state distributions of the local tracks. If the tracks have similar state descriptions then the integrand in equation (2.29) will be large, thus resulting in a high likelihood. On the other hand, if the local tracks have state descriptions

which are very different, the integrand in (2.29) will be small, resulting in a low likelihood. In equation (2.29), the function $\bar{p}(x \mid Z_{\bar{i}}^{-}, \tau)$ is identical to $p(x \mid Z_{\bar{i}}^{-}, \tau)$, the state distribution for track τ, when the track τ has a nonempty restriction at the node $\bar{i}$. When this is not the case, i.e., the track τ has not been detected yet at $\bar{i}$, the function $\bar{p}$ is scaled by the expected number of undetected targets and is no longer a probability distribution. It represents some kind of density for undetected targets.

Equation (2.30) computes the expected number of undetected targets by fusing the local track state distributions of the undetected targets. Equation (2.33) is the fusion formula for the global track state distribution. Note that it has the same form as (2.13). This is not at all surprising since given a particular track, computing the state distribution of the target is the usual estimation problem. Thus the fusion algorithm for distribution estimation is an integral part of fusion for multitarget tracking.

2.4 CONCLUSION

In this section, we have described the results of our research on information fusion for multitarget tracking. We have identified two main problems in information fusion assuming arbitrary communication. The first is how to generate meaningful tracks and hypotheses starting from a set of local tracks and hypotheses. The second is how to compute the statistics on these tracks and hypotheses when the local quantities may contain common information due to past communication.

We have developed an abstract model of the DSN in terms of the information graph. Using this graph, algorithms for information fusion have been developed. The two problems of hypothesis formation and evaluation all require keeping around histories of the tracks and hypotheses in the system. Using this history, the fusability of tracks and hypotheses can be determined. At the same time, any common information shared by their statistics can be identified so that it would not be double-counted. When specialized to broadcast communication, we can show that the general fusion algorithms for arbitrary communication reduce to those developed in the previous project.

The hypothesis formation algorithms for fusion do not depend on the target models. For hypothesis evaluation, we have assumed that the targets are static or that their motions may be approximated by "deterministic" process models. When the target models are assumed to be general Markov processes, the hypothesis evaluation algorithms have the same form as in (2.28) to (2.33). However, the state of a track would have to be a trajectory sampled at various times and computing its probability distribution would be difficult. Thus the difficulty of extending the results to treat general Markov models is more related to implementation issues. On the other hand, as long as the target motion is fairly regular, the deterministic process models we have assumed may be quite adequate.

3. TRACKING AND CLASSIFYING STRUCTURED TARGETS

This section covers our algorithm development efforts on so-called structured targets as discussed in the proposal. The treatment of dissimilar sensors is related to this task and is partially discussed in this section.

3.1 INTRODUCTION

By structured targets, we may mean two different concepts in multitarget tracking:

(1) targets with structured states, and
(2) structured sets of targets.

Since each individual target may be represented on an individual target space, the above concept (2) is one-level higher than (1). In a model based on the above concept (1), targets are still treated as individual objects although correlation among them can be considered and targets may be governed by a common state as a group of targets. This kind of models is necessary, when a multilevel identification process for each target is used or when a target has structured features. Such issues are related to the problem of treating dissimilar sensors which generate measurements corresponding to different levels of the structured target state space.

On the other hand, the concept (2) is essential when targets are in fact organized and structured in units at various levels. A typical example can be found in military units such as army → division → regiment → battalion → company, etc., in the military hierarchy. In such a case, the number of targets is typically very large and, if they are treated as independent objects, we may not be able to assess a global situation based on the outputs from any reasonably functioning target tracking system. This is so because, since grouped targets are

usually closely spaced, the data-to-data association (or scan-to-scan correlation) may become very difficult with limited computational resources. This difficulty may be overcome only when the unit structure of targets is understood and taken into account in a tracking system. Moreover, the global assessment of all the targets as a single structured object is itself an important task in many applications.

At least at the conceptual level, there is no difficulty in treating targets with structured states and dissimilar sensors in the sense that the general theory of multitarget tracking already developed through this project provides an integrated treatment of these subjects. Therefore, in the following subsections, we will mostly discuss structured sets of targets. Our emphasis is the development of a general theory upon which we may produce effective algorithms in many different applications. This should serve also as a basis for developing distributed algorithms. Section 3.2 discusses the different and the common aspects of the two different concepts, (1) and (2). In Sections 3.3 and 3.4, we will present our first-cut analysis on structured sets of targets. An algorithm is derived for two-level structured targets, i.e., tracking groups of targets. The future direction of our algorithm development effort will be discussed in the concluding section 3.5.

3.2 STRUCTURED TARGETS

As mentioned before, structured targets may connote two different concepts, each of which is discussed in the following two subsection.

3.2.1 Targets with Structured States

A typical example of a target with a structured state space may be seen in an ocean surveillance problem in which target classification is included in the tracking task. In such a case, each target i may be represented by state (x_i^c, x_i^d); x_i^c is the continuous part representing its geolocation attributes such as position, velocity, etc. x_i^d is the discrete component for its classification. The space for component x_i^d is structured and is typically represented by means of an

arborescent (tree-like) ordered set of nodes with different levels. A typical set of such levels consists of motion group → nationality → category → type → class → identification, as illustrated in Figure 3-1. In Figure 3-1, the classification tree is only partially shown and there are presumably many subtrees which are not shown. The *a priori* and *a posteriori* probability distributions on a classification tree are those on terminal nodes. The probability of an intermediate node n can be defined as the sum of all the terminal nodes which are successors of n. For any level l in the classification tree, there is a unique predecessor node given a terminal node. Therefore it is very straightforward to treat sensors with measurements at different levels. Namely each sensor observing level l nodes in the target state tree can be characterized by two transitions, i.e., one from a terminal node to a node at level l and the other from a node level l to a measurement. The first one is always with probability one while the second can be specified by the sensor characteristics.

Therefore there is no theoretical difficulty to treat targets with structured states and dissimilar sensors with measurement at different levels. In practice, however, classification trees such as the one illustrated in Figure 3-1, may be very complicated and the number of terminal nodes may be simply too many to handle in a straightforward way. In such a case, we need additional tools to effectively store and update the probability distributions on the entire terminal nodes. In [3] and [4], a set of procedures to solve such problems is shown by means of an example of ocean surveillance. Many of hypothesis management procedures devised for controlling data-to-data hypotheses (e.g., those described in [3] and being developed in the current project) can be extended to provide useful tools, e.g., hypothesis pruning, hypothesis combining and clustering. Furthermore effective representation of probability distributions must be developed in order for such management systems to work effectively. For example, track state distributions of tracks may have different representations depending on their status. Distributed processing on different levels may also be an effective procedure. For such a procedure, the theory on the distributed multitarget tracking described in earlier sections is directly applicable.

As seen in Figure 3-1, in some classification trees, all the terminal nodes are the identification (name) of each target. In other words, the identification of each target is known *a priori* and the ultimate goal of each classification is to

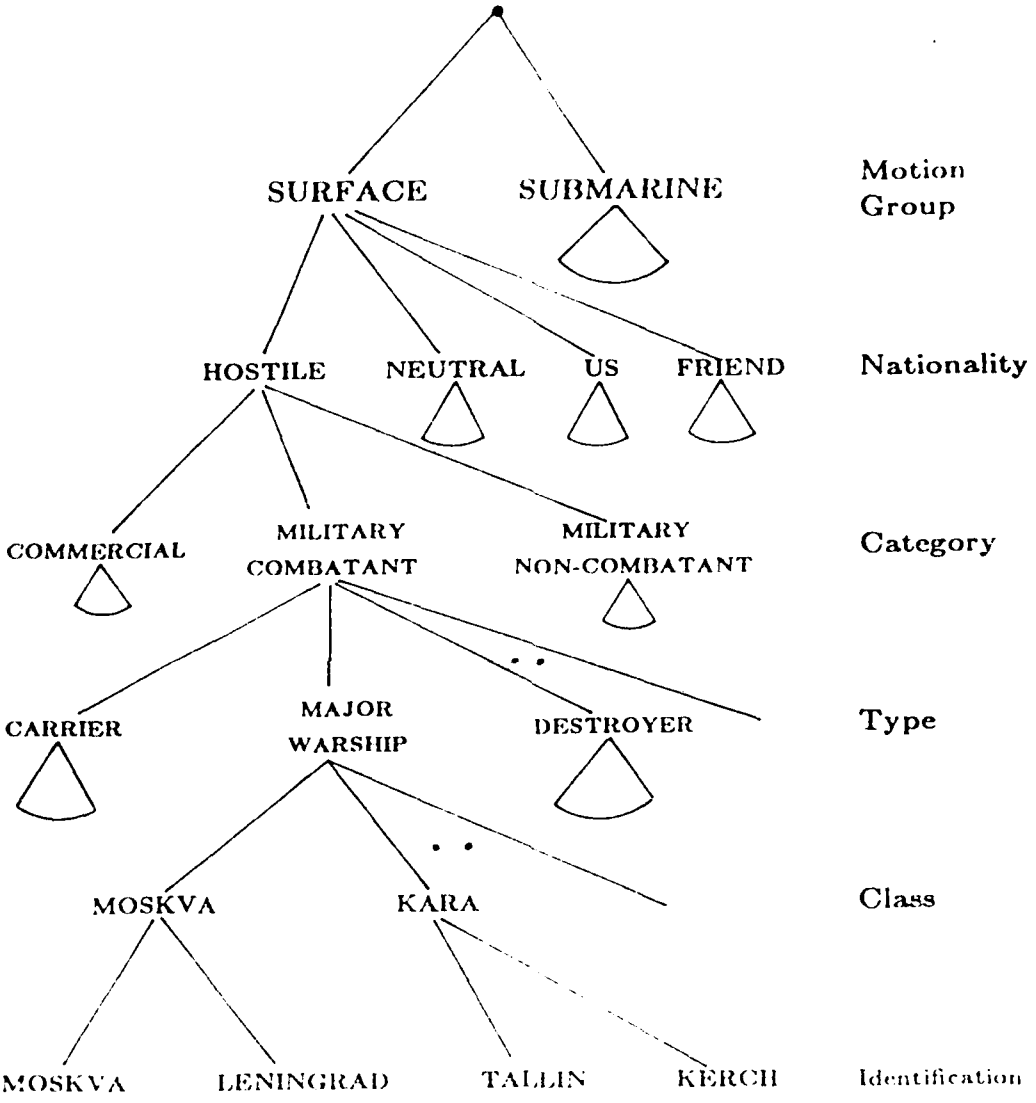


Figure 3-1: Structured State Space for a Target

determine each identification. This indicates that targets are in theory not independent from each other. For example, suppose it is known that there are only two types of targets, A and B, and that there are 10 type-A targets and 5 type-B targets. Then any hypothesis indicating more than 10 type-A targets is impossible and, if a hypothesis has fifteen tracks five of which are classified as type B, the remaining ten tracks must originate from type-A targets. However, in many cases, tracking and classification of targets can be performed quite effectively with independence assumptions and some external manipulation to take care of the dependence among targets. On the other hand, if the total number of targets of interest is small and each of them is identified *a priori*, the problem formulation based on targets with *a priori* identification, i.e., an algorithm with *a priori* tracks and hypotheses, may be more appropriate than that based on targets without *a priori* identification.

3.2.2 Structured Sets of Targets

A typical example of a structured set of targets is shown in Figure 3-2 in which a division in an army is shown in a simplified way. Depending on the type of the division, the composition and the number of subordinates, i.e., battalions have a certain pattern. The same kind of dependence is also present in the relationship between the subsequently lower levels. This kind of structure produces another dimension to the multitarget tracking problems. There are only very few theoretical results on tracking and classification of structured sets of targets. Besides a few documents referred in [5], we can only refer to a couple of technical references, [6] and [7], both of which are concerned with two-level tracking, i.e., tracking of groups of targets, but treat issues pertaining multiple groups in a rather ambiguous manner. On the other hand, AI (Artificial Intelligence) -type approaches were used in much more complicated environments in [8] and [9] which are concerned with ocean surveillance and battlefield unit identification, respectively. [8] uses a single-hypothesis propagation combined with a backtracking-like recovery scheme while [9] adopts a multi-hypothesis approach. The systems described in [8] and [9] may be viewed as hierarchical systems which may be illustrated as in Figure 3-3. The procedures represented by upward arrows are often called *bottom-up* or *induction* processes and those represented by downward arrows *top down* or *deduction* processes.

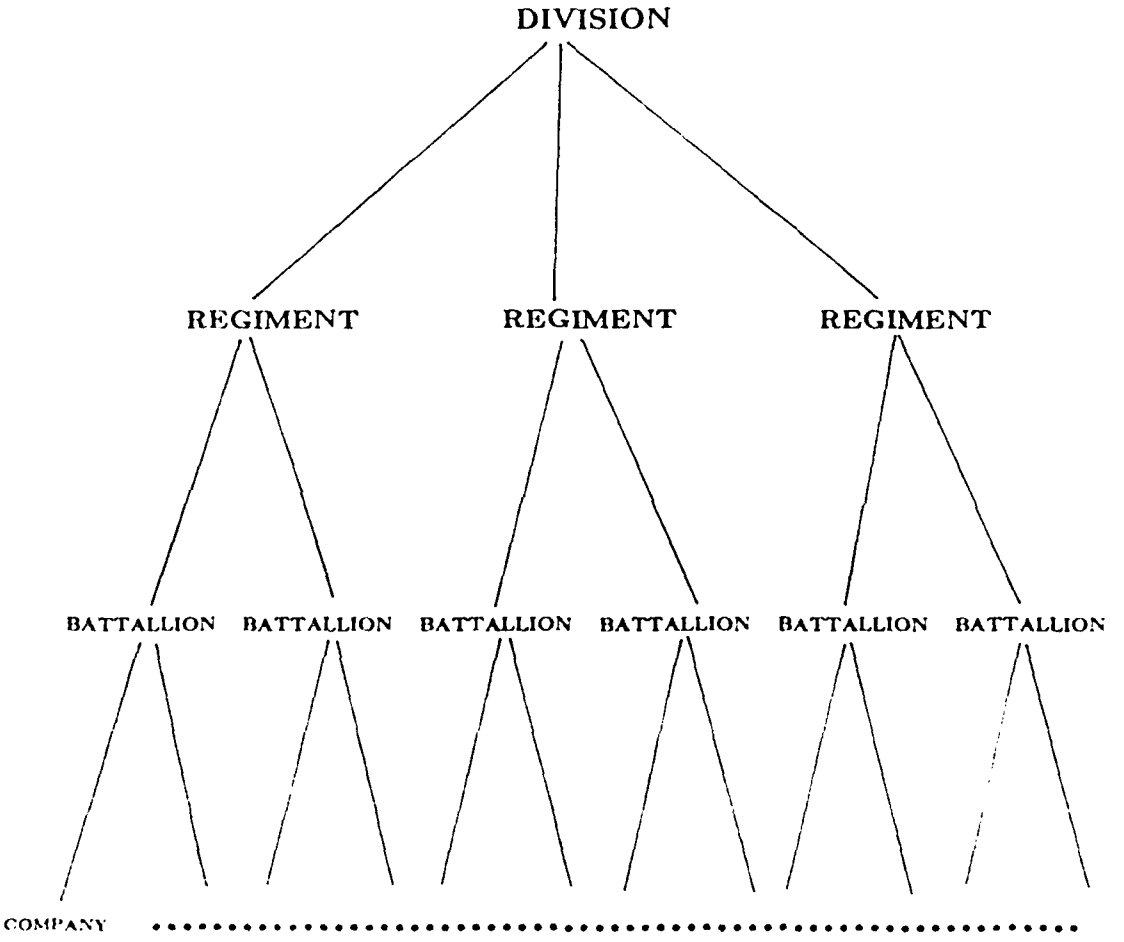


Figure 3-2: Structured Sets of Targets

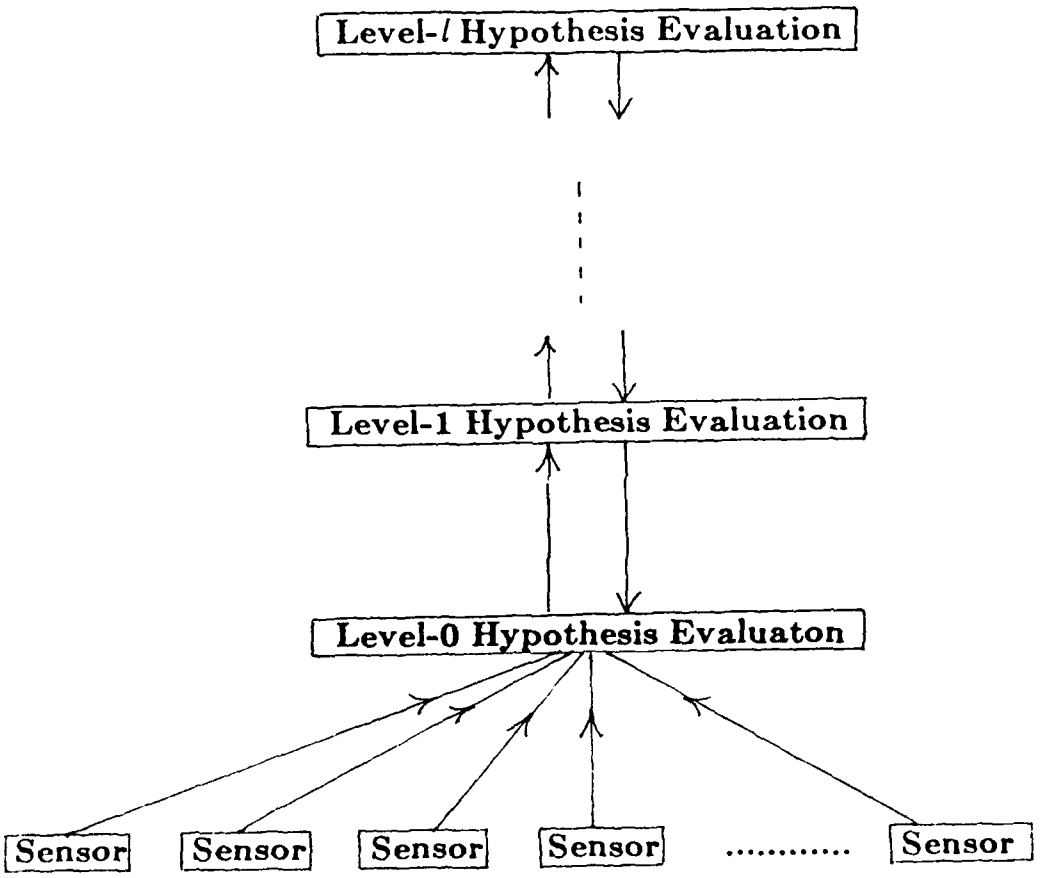


Figure 3-3: Hierarchical Hypothesis Evaluation

While the decomposition illustrated by Figure 3-3 is certainly a key to successful implementation of the systems described in [8] and [9], each hypothesis evaluation cannot be performed independently in general. For example, in tracking groups of targets, we must hypothesize possible group formation from input data while, at the same time, the states as a group must be determined and then the estimation of the states affects the evaluation of lower level hypotheses. Even if the bottom-up/top-down updating is clearly defined, iterations may be necessary for such processes to converge. Moreover, in some cases, a simple bottom-up type process may easily be overwhelmed by combinatorics. Therefore, at least for the few lower levels, we may need an integrated approach rather than a decomposition approach taken in [8] and [9]. In the subsequent subsections, we will try to establish a first-cut analysis which treats the whole structure of targets in an integrated manner.

3.3 PROBLEM FORMULATION

This subsection defines a formalism for the tracking of structured sets of targets with an arbitrary number of levels. As a first-cut analysis, we will propose a rather simplistic view.

3.3.1 A Model for Structured Sets of Targets

When we focus on each node in Figure 3-2 and its immediate successors rather than the whole picture, we notice the tree is composed of building blocks each of which has the same structure. Such a building block can be identified with a structure of a state representing a group of targets, as shown in Figure 3-4a. In tracking and classifying a group of targets, the totality of targets can be represented by (1) [level 1] the total number of targets plus a common target state component for the group, and (2) [level 0] the states of individual targets. (1) is one-level higher than (2) since (2) cannot be defined unless the number of targets is given by (1). This structure can be extended to the cases where multiple groups of targets are present. Such a case may be represented by a tree which may be illustrated in Figure 3-4b. Each level of nodes in Figure 3-4b represents: (1) [level 2] the total number of groups plus a common state

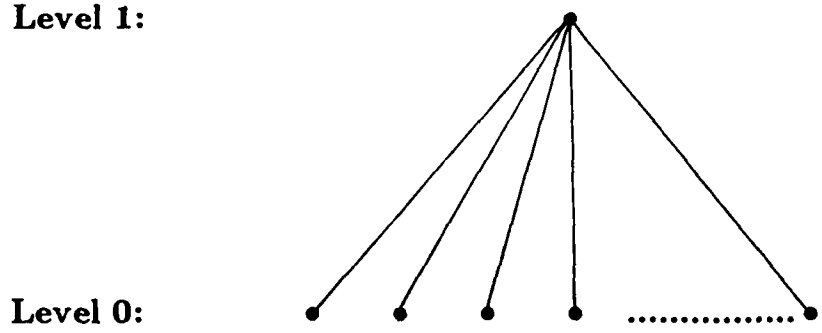


Figure 3-4a: Single-Level Targets

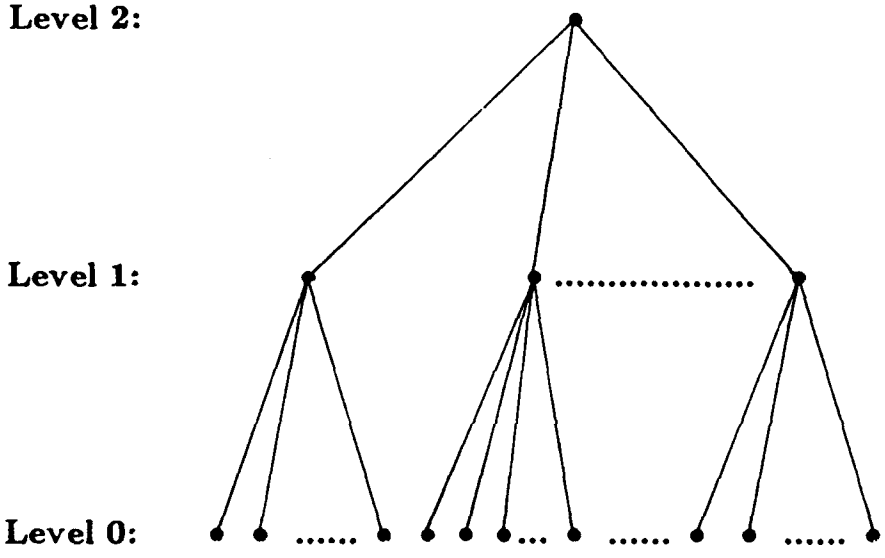


Figure 3-4b: Two-Level Targets

Figure 3-4: Single-Leval and Two-Level Targets

component for all the groups, (2) [level 1] the states of individual groups including, for each group, the number of targets in the group and a common state component for all the targets in the group, and (3) [level 0] the states of individual targets in each group.

This approach can be extended to an arbitrary level l of structures. We call such a structure a *level-l target structure* or simply a *level-l target*. As seen in Figure 3-4, when a tree represents a level-l target structure, the nodes in the tree can be labeled as level 0, level 1, ..., level l. There is always only one node at the highest level, i.e., level l. The nodes at the lowest level, i.e., level 0, represents the set of all the *targets* which we may call *level-0 targets*. In a formal description, we define a *level-l' state* for a *level-l' target* i as

$$x_i^{(l')} = (N_i^{(l')}, x_{i0}^{(l')}, x_{i1}^{(l'-1)}, \ldots\ldots, x_{iN_i^{(l')}}^{(l'-1)}) \tag{3.1}$$

where $N_i^{(l')}$ is the number of the *level-$(l'-1)$* targets in the level-l' target i, $x_{i0}^{(l')}$ is the state component common to all the level-$(l'-1)$ targets contained in level-l' target i, and each $x_{ij}^{(l'-1)}$ is the state of the j-th level-$(l'-1)$ target. Unless $l'=1$ in (3.1), every $x_{ij}^{(l'-1)}$ is defined similarly with l' being replaced by $l'-1$. When $l'=l$, there is no need to use index i in (3.1). Each level-l' target when $l'<l$ is therefore indexed as

$$i = (i_{l-1}, \ldots\ldots, i_{l'}) \ . \tag{3.2}$$

According to an alternative view of this approach, we are first given a set of targets, then a partition of the targets into multiple groups, then a partition of the groups into multiple super-groups, and so forth. In other words, a level-l' target is an element of a partition of the set of all the level-$(l'-1)$ targets. The partition is a trivial one when $l'=l$. In typical battlefield units as shown in Figure 3-5, each unit has its headquarter (division command post (DCP), regiment headquarter (RH), battalion headquarter (BH), etc.) besides its subordinates (R = a regiment, B = a battalion, C = a company, etc.). These headquarters may be considered either (1) as a part of the common state of each level-l' target or (2) as special targets which do not have any subordinate. When we adopt the latter consideration, we may simply extend each headquarter node to the lowest level, i.e., level 0. As mentioned before, as a first-cut analysis, we ignore such problems. There will be no problem in rectifying the formulation to treat headquarters in appropriate ways in the future.

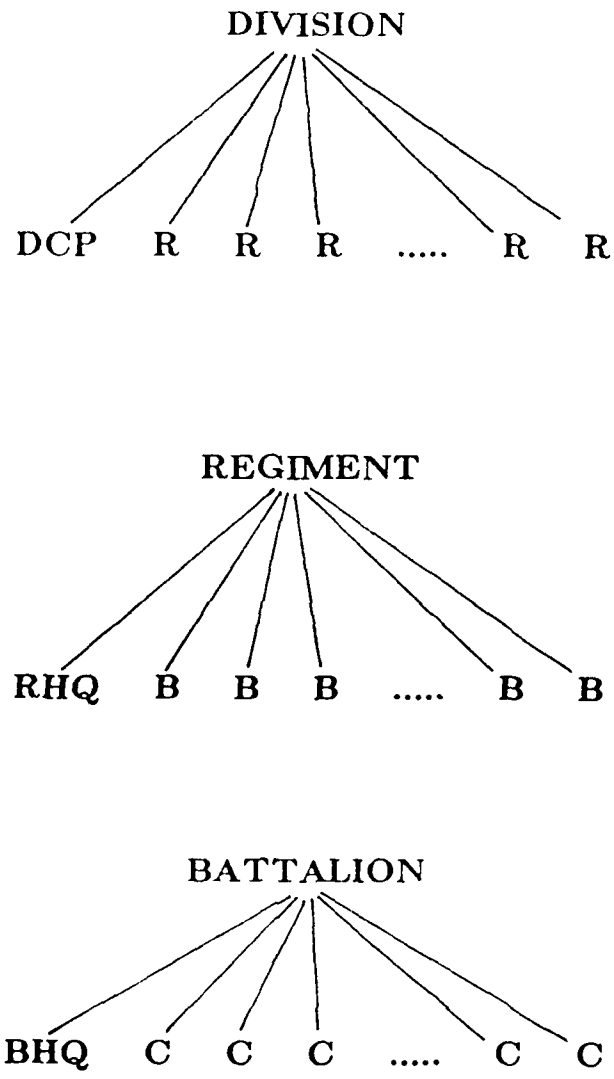


Figure 3-5: Battlefield Units

3.3.2 Sensor Models and Multi-Level Tracks and Hypotheses

We can extend our target-sensor model for multitarget tracking from single-level cases to multi-level cases in a rather straightforward way as follows: Let S be a finite set of *sensors* which observe the targets. For each sensor s, the *measurement value space* Y_s in which measurements from sensor s take values is assumed to be a hybrid space. Each output from sensor s is a *data set* $(y_1,.....y_m,m,t,s)$ which is an element of

$$\bigcup_{m=0}^{\infty} \bigcup_{s\in S} (Y_s)^m \times \{m\} \times [t_0,\infty) \times \{s\}$$

and represents m measurements, $y_1,.....,y_m$, generated by sensor s at time t. (t_0 is the time before which no sensor outputs any data set.) A collection of data sets available up to a certain time is called a *cumulative data set.* We assume that all the data sets are indexed by positive integers as $z(1),z(2),.....$, where

$$z(k) = ((y_j(k))_{j=1}^{N_M(k)}, N_M(k), t_k, s_k) \tag{3.3}$$

for each positive k such that $t_k \leq t_{k'}$ whenever $k < k'$. A *cumulative measurement set up to k* is defined as

$$J^{(k)} = \bigcup_{k'=1}^{k} \{1,.....,N_M(k')\} \times \{k'\} \tag{3.4}$$

For the sake of simplicity, we assume that possible origins of measurements in any data set are only level-0 targets. Let I_T be the set of level-0 target indices. For each data set k, we assume an *assignment function* A_k defined on a subset of the level-0 target index set I_T taking values in $J(k) \triangleq \{1,.....N_M(k)\}$. When $j = A_k(i_{l-1},.....,i_0)$, we say level-0 target $(i_{l-1}, \ldots, i_0)$ is *detected by* sensor s_k at time t_k and *generates* the j-th measurement, or the j-th measurement *originates from* level-0 target $(i_{l-1}, \ldots, i_0)$. With the no-split/no-merged measurement assumption, such an A_k is a well-defined one-to-one function. Then, given a cumulative data set, we can define the trace of a level-l' target in it in the form of a subset of the cumulative measurement index or a collection of such subsets at the given level. We call any possible realization of such a trace a *level-l' track.* Thus a subset of the measurement index set is a level-0 track if it contains at most one measurement index set for each data set. A level-l' track is a collection of nonoverlapping level-$(l'-1)$ tracks. A *level-l' hypothesis* is then a

collection of nonoverlapping nonempty level-l' tracks and hypothesizes all the set of measurements originating from level-l' targets. According to this definition, a level-1 track is also a level-0 hypothesis, and *vice versa*, although its interpretation as a track is completely different from that as a hypothesis.

Multi-level hypotheses defined above may be illustrated in Figure 3-6 in which $l=3$ and a level-2 hypothesis is represented by a tree depicted by solid lines. In Figure 3-6, the level-2 hypothesis consists of two level-2 tracks each of which hypothesizes a group of detected groups of targets, $\{\{\tau_1,\tau_2\},\{\tau_3,\tau_4\}\}$ or $\{\{\tau_5\},\{\tau_6,\tau_7\}\}$, where τ_1 to τ_7 are level-0 tracks each of which hypothesizes a detected level-0 target. Given such a hypothesis, we must further hypothesize the existence of undetected targets and the overall structure, as shown in Figure 3-6 by broken lines. The process to group given level-0 tracks τ_1 to τ_7 in a level-0 hypothesis into a level-1 hypothesis and then into a level-2 hypothesis can be viewed as a *bottom-up* procedure. While the process to add hidden targets and to complete the overall structure can be viewed as a *top-down* procedure. The evaluation of hypotheses may not be, however, decomposed in such a manner. The discussion of hypothesis evaluation in a general level-l case may be very complicated. Therefore, in the following sections, we will restrict ourselves to the cases where $l=2$, i.e., where tracking of multiple groups of targets is concerned.

Remark: In the above discussion, we only considered the cases where each measurement from each sensor is based on a level-0 target. The definitions of tracks and the hypotheses may be altered so that measurements from different levels may be treated. At this moment, however, the exact form of the appropriate modification is not very clear.

3.4 EVALUATION OF TWO-LEVEL HYPOTHESES

In this subsection, we will extend our general theory of multitarget tracking from single-level cases to two-level cases, i.e., tracking multiple groups of targets. The issues pertaining to implementation will be briefly discussed in terms of an example.

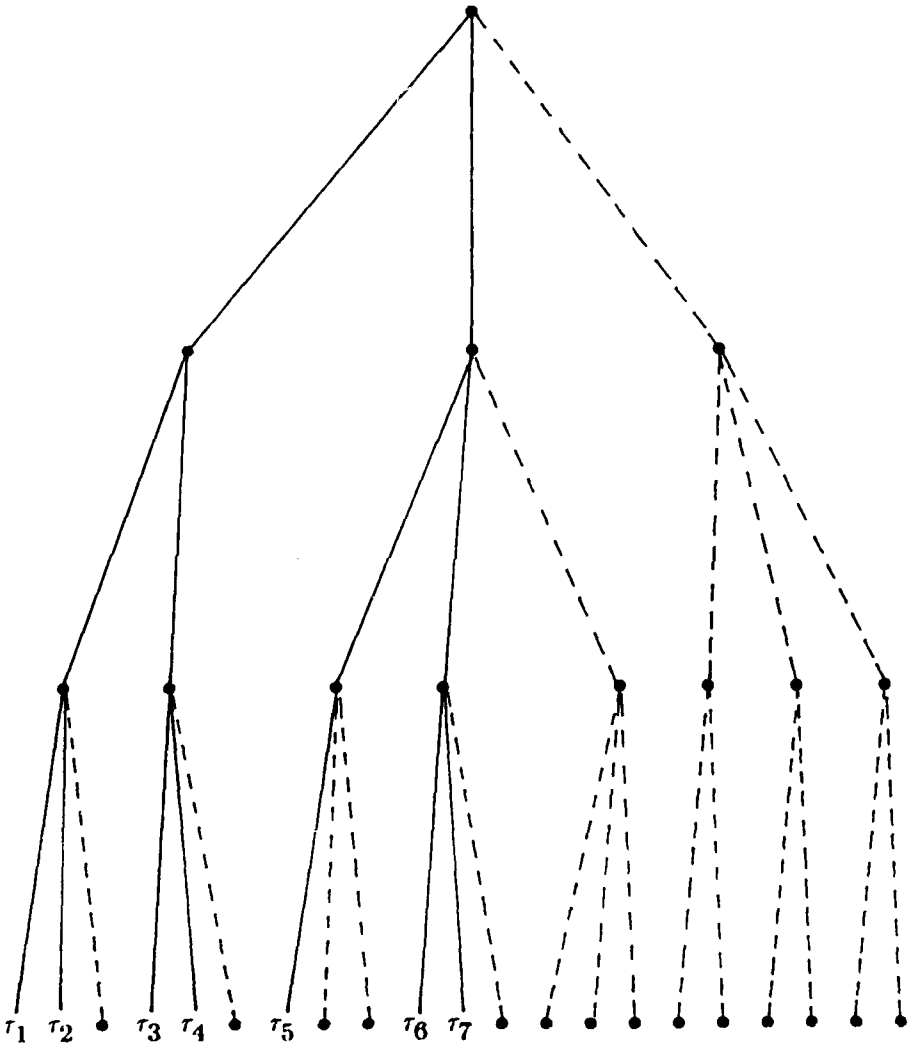


Figure 3-6: Multi-Level Hypothesis

3.4.1 Two-Level Multitarget Tracking

When the target structure level is two, i.e., $l=2$, the overall target state can be written as

$$x = (N_G, x_0, x_1, \ldots\ldots, x_{N_G}) \tag{3.5}$$

where N_G is the total number of groups, x_0 is the state component common to all the groups, and each x_i is the i-th group's individual state. Each x_i is therefore in form of

$$x_i = (N_i, x_{i0}, x_{i1}, \ldots\ldots, x_{iN_i}) \tag{3.6}$$

where N_i is the number of (level-0) targets in group i, x_{i0} is the state component common to all the targets in group i, and x_{ij} is the individual states of the j-th target in group i. Let the level-1 target index set be $I_G = \{1, \ldots\ldots, N_G\}$ and the level-0 track index set be $\bigcup_{i=1}^{N_G} \{i\} \times \{1, \ldots, N_i\}$. Then the trace of level-0 target (i_1, i_0), i.e., the i_0-th target in the i_1-th group, in a cumulative data set up to k is

$$T_k^{(0)}(i_1, i_0) = \{(j, k') \mid j = A_{k'}(i_1, i_0), 1 \le k' \le k\} \quad . \tag{3.7}$$

The trace of level-1 target i_1 is then

$$T_k^{(1)}(i_1) = \{T_k^{(0)}(i_1, i_0) \mid 1 \le i_0 \le N_{i_1}\} \quad . \tag{3.8}$$

Then a level-0 hypothesis is a possible realization of

$$\Lambda_k^{(0)} = \{T_k^{(0)}(i_1, i_0) \mid T_k^{(0)}(i_1, i_0) \neq \emptyset, (i_1, i_0) \in I_T\} \tag{3.9}$$

and a level-1 hypothesis is a possible realization of

$$\Lambda_k^{(1)} = \{T_k^{(1)}(i_1) \mid T_k^{(1)}(i_1) \neq \{\emptyset\}, i_1 \in I_G\} \quad . \tag{3.10}$$

We can extend the concept of *target-to-track hypothesis* from single-level tracking to two-level tracking as follows: A *level-1 target-to-track hypothesis* is a possible realization of a one-to-one random function from $\Lambda_k^{(1)}$ to I_G defined by

$$\Omega_k^{(1)}(T_k^{(1)}(i)) = i \tag{3.11}$$

and a *level-0 target-to-track hypothesis* is a possible realization of a one-to-one function from $T_k^{(1)}(i)$ to $\{1, \ldots\ldots, N_i\}$ (given $T_k^{(1)}(i)$) defined by

$$\Omega_k^{(0)}(T_k^{(0)}(i_1,i_0);T_k^{(1)}(i_1)) = i_0 \quad . \tag{3.12}$$

As in the theory of single-level multitarget tracking, whenever we must distinguish a realization of $\Lambda_k^{(l)}$ from that of $\Omega_k^{(l)}$, we call the former *data-to-data hypothesis.*

3.4.2 General Results

We will derive a recursive formula for calculating each level-1 hypothesis. The results are an extension of the single-level tracking results. For the rest of this section, we make the standard set of assumptions: (1) Targets are interchangeable *a priori.* (2) The data sets are conditionally independent given the target states. (3) The assignment functions are totally random. The first step is a straightforward recursive formula

$$P(\Lambda_k^{(1)} \mid Z^{(k)}) = \frac{P(Z^{(k)},\Lambda_k^{(1)} \mid Z^{(k-1)},\Lambda_{k-1}^{(1)})}{P(Z^{(k)} \mid Z^{(k-1)})} P(\Lambda_{k-1}^{(1)} \mid Z^{(k-1)}) \quad . \tag{3.13}$$

The numerater on the RHS of (3.13) can be expanded in a way similar to that used for the single-level tracking (as described in [1] and [2]) and yields

$$P(\Lambda_k^{(1)} \mid Z^{(k)}) = \frac{P(\Lambda_{k-1}^{(1)} \mid Z^{(k-1)})}{P(Z^{(k)} \mid Z^{(k-1)})} \tag{3.14}$$

$$\sum_{N_G=\#(\Lambda_k^{(1)})}^{\infty} \frac{(N_G-\#(\Lambda_{k-1}^{(1)}))!}{(N_G-\#(\Lambda_k^{(1)}))!} P(N_G \mid \Lambda_{k-1}^{(1)},Z^{(k-1)})$$

$$\sum_{N^G \in \Lambda_k^{(1)}} \prod_{\tau} \frac{(N_{\Omega_k^{(1)}(\tau)}-\#(\overline{\tau}))!}{(N_{\Omega_{k-1}^{(1)}(\tau)}-\#(\tau))!} P(N^G \mid N_G,\Omega_k^{(1)},\Lambda_{k-1}^{(1)},Z^{(k-1)}) \mathbf{L}_k(z(k) \mid A_k,N^G,Z^{(k-1)})$$

where

$$N^G = (N_G,N_1,\ldots\ldots,N_{N_G}) \tag{3.15}$$

and

$$\mathbf{L}_k(z(k),A_k,N_G \mid Z^{(k-1)}) \propto \frac{N_{FA}(k)!}{N_M(k)!} \tag{3.16}$$

$$\int P((y_j(k))_{j=1}^{N_M(k)} \mid A_k, N_M(k), x(t_k), N^G)\, P_M(N_M(k) \mid I_{DT}(k), x(t_k), N^G)$$

$$P(I_{DT}(k) \mid x(t_k), N^G)\, P(dx(t_k) \mid N^G, \Omega_k^{(0)}, Z^{(k-1)}, \Omega_k^{(1)}) \quad .$$

The updating formulae for $P(dx(t_k) \mid N^G, \Omega_k^{(0)}, \Lambda_k^{(1)}, Z^{(k)})$, $P(N^G \mid N_G, \Omega_k^{(1)}, \Lambda_k^{(1)}, Z^{(k)})$ and $P(N_G \mid \Lambda_k^{(1)}, Z^{(k)})$ can be derived in a similar way.

3.4.3 I.I.D.-Poisson Groups

In single-level tracking, an appropriate set of independence assumptions enables us to reduce a general form into a more implementable form. We will repeat such a process for two-level multitarget tracking. The additional assumptions are as follows:

[1] Given the number N_G of groups, the group states tuple $(x_i)_{i=1}^{N_G}$ is a system of independent Markov processes which share common joint probabilitites. Thus the state component x_0 common to all the groups is ignored. The number N_G of groups has a Poisson distribution with mean ν_0.

[2] Each $x_i = (x_{i0}, x_{i1}, \ldots, x_{iN_i})$, given N_i, is a stochastic process such that $(x_{ij})_{j=1}^{N_i}$ is a system of interchangeable Markov processes.

[3] The detection is target-wise independent, i.e., the detection of target (i_1, i_0) depends only $(x_{i_1 0}, x_{i_1 i_0})$ and we have

$$P(I_{DT}(k) \mid x(t_k), N^G) = \tag{3.17}$$

$$\prod_{(i_1, i_0) \in I_T} p_D(x_{i_1 0}, x_{i_1 i_0} \mid k)^{\delta(i_1, i_0)} (1 - p_D(x_{i_1 0}, x_{i_1 i_0} \mid k))^{1 - \delta(i_1, i_0)}$$

with a common *detection probability function* p_D.

[4] Measurement errors are also target-wise independent, i.e., the value of a measurement originating from a target (i_1, i_0) is correlated only to

$(x_{i_1 0}, x_{i_1 i_0})$. The number of false alarms and their values are independent of the targets and from data set to data set. Thus we have

$$P(N_M(k) \mid I_{DT}(k), x(t_k), N^G) = p_{N_{FA}}(N_M(k) - \#(I_{DT}(k)) \mid k) \tag{3.18}$$

and

$$P((y_j(k))_{j=1}^{N_M(k)} \mid A_k, N_M(k), x(t_k), N^G) = \tag{3.19}$$

$$\left(\prod_{(i_1,i_0)\in Dom(A_k)} p_M(y_{A_k(i_1,i_0)}(k) \mid x_{i_1 0}(t_k), x_{i_1 i_0}(t_k), k)\right)\left(\prod_{j\in J_{FA}(k)} p_{FA}(y_j(k) \mid k)\right)$$

with a *number-of-false-alarm probability function* $p_{N_{FA}}$, a *target-state-to-measurement transition probability density function* p_M, and a *false-alarm-value probability density function* p_{FA}.

Under these assumptions, we can derive results which are very analogous to those in single-level tracking (described in [1] and [2]) and are summarized as follows: (1) Given a level-1 target-to-track hypothesis, the posterior distributions of the group states $(x_i)_{i=1}^{N_G}$ are independent, (2) the posterior distribution of undetected groups becomes Poisson, and (3) the hypothesis evaluation can be reduced to the evaluation of level-1 track-to-measurement likelihood as

$$P(\Lambda_k^{(1)} \mid Z^{(k)}) = \frac{P(\Lambda_{k-1}^{(1)} \mid Z^{(k-1)})}{P(Z^{(k)} \mid Z^{(k-1)})} \frac{\exp(\nu_k - \nu_{k-1})}{N_M(k)!} \tag{3.20}$$

$$L_{FA} \prod_{\tau\in\Lambda_k^{(1)}} L(y\{\tau \mid k\} \mid Z_\tau^{(k-1)})$$

where L_{FA} is the false alarm likelihood,

$$y\{\tau \mid k\} = \{y_j(k) \mid (j,k)\in\bigcup\tau\} \tag{3.21}$$

is the set of measurements assigned to level-1 track τ and $L(\cdot \mid Z_\tau^{(k-1)})$ is the level-1 track-to-measurement likelihood. The forms of the above likelihood functions are very similar to that of the hypothesis evaluation formula for the single-level tracking of dependent targets.

3.4.4 An Example

A straight forward extension of single-level tracking to two-level tracking is possible using the results shown in the previous two subsections. In two-level tracking, however, the combinatorial problem is even more severe, which may make a straightforward extension of single-level trackers infeasible in many applications. For this reason, we may have to develop new techniques for overcoming the additional combinatorial burden inherent to two-level tracking. In this subsection, we will discuss this aspect of the problem in terms of a simple example.

We consider tracking of groups of ground vehicles moving on a road network. By a two-step transformations to take care of (1) the route selection by each group and (2) the curvature of each road segment, the problem can be reduced to that of tracking groups of targets moving on a straight line. Let u_i be the 1-dimensional position of the lead vehicle of the i-th group and v_i be its velocity. Then the position and the velocity of the j-th vehicle in group i can be modeled as

$$u_{ij} = u_i - (j-1)c_i v_i + \xi_{ij} \tag{3.22}$$

and

$$v_{ij} = v_i + \eta_{ij} \tag{3.23}$$

where $c_i v_i$ is the expected distance between two vehicles in group i, ξ_{ij} and η_{ij} represent randomness in position and velocity of each vehicle in the group. We assume that the randomness can be modeled by independent gaussian random variables. The group dynamics are then assumed to be a simple almost constant velocity model with an appropriate white gaussian driving noise. Thus we may have a very simple target model in which the state component common to all the targets in group i is

$$x_{i0} = (u_i, v_i, a_i) \tag{3.24}$$

where a_i is a discrete variable representing the *type* of group i. The individual target state of the j-th target in group i is then simply its *type* a_{ij}.

For each possible type of group, we assume that we have a sufficient number of *templates* of the group including composition of different types of vehicles and their order when moving on the road. Each template can be represented by

$$e = (a, N, b_1, \ldots, b_N) \tag{3.25}$$

where a is the type of a group, N is the number of vehicles in the group and b_i is the type of the i-th vehicle in the group. Therefore the level-1 track distribution, i.e., the group state distribution, is a distribution on $(-\infty,\infty)^2 \times E$, where E is the set of all the templates. In general, we may assume at least in an approximated sense the independence of motion from the type component as $P(du, dv, de) = P(du, dv)P(de)$.

When a data set is received from a sensor, each group hypothesis is given a set of measurements which may be associated to it. Then the set of measurements is ordered linearly and, for each template, the level-1 track-to-measurement likelihood is calculated after template-to-measurement matching as shown in Figure 3-7. In such a process, we must use a very effective method for determining a likely level-1 track-to-measurement assignment. For example, for each template, we first estimate the most probable distance between targets based on the velocity estimate and then spread the vehicles in the template accordingly. Then, by an effective assignment algorithm, we can find a feasible assignment between the given set of measurements and the vehicles in the template. After determining the assingment, we can calculate the level-1 track-to-measurement likelihood.

3.4.5 Distributed Hypothesis Formation and Evaluation

As shown earlier in this report, distributed hypothesis formation is a process of creating a logically consistent set of hypotheses from a collection of local sets of hypotheses. This process amounts to the consistency checking on the overlapped pieces of information in the past. It is also determined purely by the definitions of tracks and hypotheses and independent of their probabilistic nature. Therefore it is expected that we may extend the single-level tracking results to the two-level or in general level-l tracking cases. The results may be a

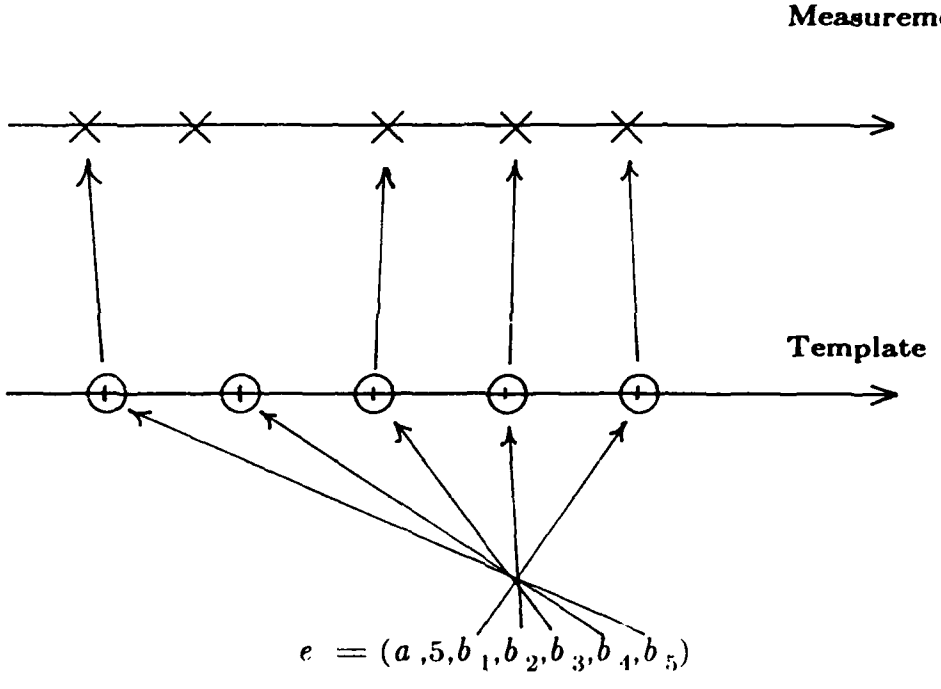


Figure 3-7: Template-to-Measurement Matching

similar type of consistency checking on the predecessors of tracks and hypotheses. However, although the final results are fairly simple in single-level tracking cases, complicated steps were necessary to derive implementable results. It is hence expected that the logical arguments involved in two-level tracks and hypotheses may well be very complicated.

On the other hand, distributed hypothesis evaluation involves the distributed estimation and is highly dependent on the structure of the global hypothesis evaluation formula. In the single-level tracking cases (with the i.i.d. Poisson assumption), as shown in the appendix, the hypothesis evaluation equation is, in essence, a product of track likelihoods and each track likelihood is an integration of a product of state-to-measurement transition probability densities. Thus each track likelihood can be decomposed using distributed estimation theory. In two-level tracking cases, however, the level-1 track-to-measurement likelihood involves summation over many possible numbers of targets in each group, which may cause difficulty in decomposing the track likelihood into the independent components. We may well need a kind of aggregation of tracks and hypotheses in order to produce a workable algorithm for distributed hypothesis evaluation for the two-level tracking. The dynamic behavior of groups may also complicate the discussions.

3.5 CONCLUSION

A first-cut analysis on multitarget tracking concerning structured targets were discussed in this section. The discussions in this section are summarized as follows: (1) The treatment of targets with structured state spaces is at least theoretically straightforward. (2) The treatment of such targets in practice may, however, need several additional consideration and more aggressive hypothesis management strategies. (3) The same arguments as in (1) and (2) are also valid when dissimilar sensors with measurements at different levels are concerned. (4) Structured sets of targets may be treated in an integrated form and concepts of tracks and hypotheses can be extended from the single-level cases in a straightforward way. (5) Two-level multitarget tracking hypothesis evaluation can be done by extending the single-level tracking results. (6) Practical methods for implementing two-level hypothesis evaluation needs however further

investigation. (7) Distributed hypothesis formation and evaluation for two-level tracks and hypotheses may be possible by extending the single-level results but we need more time to clear this problem. The future efforts pertaining to the topics covered in this section may include: (1) effective implementation of single-level tracking with correlation among targets, (2) implementation of two-level multitarget tracking algorithms, and (3) development of distributed level-1 hypothesis formation/evaluation algorithms.

4. TOWARD A DSN DESIGN TESTBED

In this section we describe an architecture for a general testbed environment within which a DSN sytem may be designed, prototyped, and its performance capabilities tested on a simulation of the task domain. The major aspects of this architecture, called *Schemer*, have been implemented and used for the development of distributed systems similar to that of a DSN [1]. Judging from the success we have had with early versions of Schemer for distributed system development support we believe that a complete Schemer system can provide a development environment within which the structure of a DSN system can be evolved and the performance abilities of alternative techniques for sensor and other forms of information processing (such as the Bayesian tracking and classification algorithm being investigated in this project) can be "empirically" evaluated. Moreover, since Schemer is an extension of techniques for AI expert system construction, it also provides direct support for augmenting non-AI approaches to signal processing, information fusion (both signal and message level), resource allocation, control, etc. by combining these non-AI alogorithms with powerful AI heuristic methods for problem solving, planning, and other forms of intelligent reasoning.

Before going into details, let us give an overview of some of the important assumptions and concepts that underlie the approach being taken with Schemer. The Schemer based approach to be described in more detail below capitalizes on the notion that a distributed system like a DSN may be viewed as a kind of distributed decision making (DDM) system. That is, each node (an individual processing element) in a DSN system may be viewed as a more or less independent "decision maker" reaching conclusions on the basis of its own privately acquired information and information shared with other nodes in the overall system. This "cognitive' top level view is consistent with other work we have done using Schemer as the basis of a distributed system model [11]. In general then the Schemer system must provide a context within which to model all the following aspects of DSN system structure and function, as well as the target task domain:

1. The event stream of the task domain to which the overall system is exposed, and the part of that overall event space to which each node in the distributed system will be individually exposed;
2. The structural relations that obtain among the nodes of the DSN system, including the communication relations among the nodes and the authority or control relations among these same nodes; and
3. The performance and decision making abilities of each node;
4. The criteria according to which system performance should be judged, including specification of the global "goals" and task requirements of the DSN system and other performance criteria that constrain how these global goals should be met.

More should be said about the metaphor of each DSN node as a participating decision maker in the overall distributed system. The technology of *AI Expert System (AI/ES)* construction has evolved precisely in response to the need to provide computational models that incorporate expertise for performance and decision making in some task domain. In this sense each node of the DSN system can be thought of as a participating ES armed with its own local expertise for carrying out its particular functions in the network and for cooperating with other nodes with which this node can be in contact. Furthermore, this viewpoint exposes the importance of the idea that we should where possible incorporate available human expertise in the capabilities of a DSN node to augment other non-AI methods like GTC. Thus, the metaphor of a DSN as a network of decision making participants leads to the view of a DSN as a cooperative distributed expert system.

The central principles of the Schemer design to be presented below derive from the need to provide a testbed environment in which to explore new approaches to the design of knowledge based Expert Systems (ESs). The various alternative ES architectures that have been developed by the AI community provide suitable computational models of human expertise for performance and decision making in a wide range of domains [12]. Schemer therefore is intended to provide a framework for constructing a system out of some combination of these AI architectural techniques in combination with non-AI capabilities. Furthermore, the basic Schemer framework has been developed with an eye toward construction of *distributed Expert System networks*.

In the following discussion we will briefly sketch the requirements on testbed design, and then present our ideas for an architecture intended to meet these requirements. As noted above, this architecture can be viewed as a kind of framework for designing and protoyping distributed, cooperative expert systems that can be configured for a wide range of application domains.

4.1 REQUIREMENTS FOR A DESIGN TESTBED

The primary categories of activities that require support are DSN system design development, rapid prototyping of the DSN design, and prototype evaluation. Let us briefly review these activities and the requirements that each imposes on a system intended to support a designer engaged in each of them.

The support of design development involves two principal issues. One of these is support for specifying some candidate design for the DSN system. The other is the specification of the tasks to be performed by the DSN system and its constituent elements and a task environment in which the system will operate.

As noted above, we view the problem of specifying a DSN system design as essentially the same as that of constructing a model of a distributed expert system. To review this claim, the designer models the individual elements or participants in the DSN network by defining each as a particular expert system structure. Thus, each expert system is constructed to "model" the decision making and other capabilities that a DSN node must be able to provide in response to specific task conditions that may arise as the entire system performs its overall activities. The designer's specifications for the capabilities required of an element are modeled as the expertise of the expert system representing that DSN element.

In addition to modeling the elements of the system, the designer must also model the relationships that obtain among these elements. The principal types of relations among system elements that must be modeled are the communication relations and the control (i.e., authority) relations that exist among nodes. There are various approaches being developed by the AI community for modeling communication and control relations in a distributed system of (potentially)

intelligent agents. In other work completed at AI&DS [10], we have reviewed some of the major issues that must be addressed in dealing with these problems of distributed system communication and control. Schemer's distributed AI/ES approach is, prima facie, an excellent path modelling such applications such as a DSN. However, we note as we did earlier that the modeling capabilities of a design support environment should also support the modeling of other "unintelligent" types of system elements (e.g., a remote sensor or senor system).

The task context in which the DSN system operates must be modeled if the designer is to explicitly consider the relationship between candidate design and that task context. In order to support the DSN system designer in specifying the task requirements and task environment the DSN design testbed must contain tools for modeling the objects and events to be encountered by the DSN system as it performs its tasks, and also descriptions of the performance expected of the DSN system itself. The design testbed must provide a means of expressing scenarios and event streams that can be fed to the DSN system model for simulation, the next major capability to be provided in the design testbed.

Therefore, using the expert system model of a DSN system for simulation is the approach we propose for prototyping a candidate design. Furthermore, in order to promote rapid prototyping, it is desirable to allow the designer to conduct simulations while aspects of the total system design are incompletely or generically specified. This approach to the prototyping task requires a testbed environment which contains methods for using the models of the task context and the system itself to simulate the performance of the intended system. Moreover, it should be possible to abstractly or generically specify subsystems of the complete system and still perform at least limited simulations of system performance.

The final major aspect of our design methodology has to do with the problem of design evaluation. Designs for systems of the complexity of DSN systems do not readily yield to analytical methods. Thus, a primary value in providing the kind of simulation and rapid prototyping capability just discussed is so that methods for empirical analysis can be used where analytical approaches to design evaluation are not available. Furthermore, it is intended that these observational techniques be used as early as possible in the design development process to

promote rapid convergence on a satisfactory design. The implication of these remarks is that the design methodology we seek to instantiate should provide a mixed approach including analytic verificational techniques for those types of system and subsystem designs that admit such analyses, and observation based methods of evaluation for designs sufficiently complex that no known analytical evaluation method applies.

We can now look at the specific Schemer architecture as an approach to the construction of a DSN design testbed that meets the design support requirements just reviewed. In the next section we will present the general approach to knowledge representation provided in Schemer to handle both DSN system and task context modeling. Following that discussion we will present a description of the design of the Schemer testbed architecture and consider the kinds of tools needed to construct a DSN protoptype, run simulations with it, and evaluate its performance in terms of task specifications.

4.2 KNOWLEDGE REPRESENTATION IN THE DESIGN TESTBED

Our discussion of knowledge representation will proceed in parallel with the discussion of testbed requirements above. First, let us consider an approach to modeling a single DSN node. This will first focus on how a designer may represent the knowledge held by a node that allows it to perform its function. This will also require that we show how this node is controlled in some general framework. To this end we will next present a generic expert system architecture within which the capabilities of a specific DSN component may be modeled. Finally, we will look at the relationships among nodes specifically considering how communications and authority relations among nodes can be expressed.

After considering how to represent an individual node, we will then discuss how the task environment may be represented in the design testbed. As we will see, the representational requirements for this aspect of modeling will largely be another application of the techniques that we have developed for representing the individual DSN system nodes' internal structure.

We will defer discussion of design tools and their representation until after we have looked at the overall architecture of the design testbed. This we will do in the next section.

4.2.1 Node representation in a DSN system

We begin now by discussing our approach to modeling a node. Consider first the knowledge that may be held by a node. We need to consider the following four categories:

a. Static (i.e., relatively permanent) knowledge;

b. Situational (current context) knowledge;

c. Planning knowledge; and

d. Control knowledge.

4.2.1.1 Static knowledge

Static knowledge refers to the relatively unchanging knowledge that a node has of the objects and events in the task domain. This will include other nodes in the DSN system as well as objects external to the system. In this project and in previous research and development efforts at AI&DS, we have considered some of the basic issues that are involved in representing the types of knowledge that must be held by a node. The design conclusion that we have reached on the basis of this analysis is simple to state. We have determined that an expanded frame representation language [13] is best suited to provide the representational power and flexibility needed for modeling the static knowledge of a DSN node. (In fact our use of a frame representation is even more pervasive, but we hold further elaboration of this point until later.)

A node's knowledge of a type of object (e.g., a particular type of missile) is represented as a structured combination of attributes and, optionally, restrictions on the values of those attributes that may be held by an instance of that object type. Figure 4-1 exemplifies this types of representation. In the figure we see that the object category "Backfire_bomber" is identified in terms of specific attributes such as wingspan, top speed, etc. Each of these attributes is likely to have a specific value or a range of values (expressed as a construct known as a "constraint") which must be satisfied for any object to count as an instance of that class. It is also possible that the value of an attribute can be a complex structure. For example, certain types of aircraft are distinguishable in terms of their tail assembly configuration. Thus, the frame representation could contain an attribute called "Tail_configuration" which is itself another frame description whose attributes are the components of the tail assembly, their descriptors, and the relationships among these components.

Another important aspect of a frame representation such as the type we are considering here has to do with the notion of "inheritance". This refers to the way that the information described in one frame may be accessed as part of the description available in another related frame. The frame description of the figure has an attribute called "A_kind_of" that relates this frame to another, more general, frame for the more general class bombers. In that latter frame there are contained a number of descriptors such as "Bomb_payload: greater than 0" which serve to more generically define the features of that class. In Figure 4-1 that attribute has been "inherited" and has been given a more specific restriction.

Thus, the attributes of a frame description determine the descriptive properties of a particular class, and additionally can specify the relationship of that class to other classes. These interclass relationships provide the basis for using the knowledge represented in one frame to augment the definition/description provided by another frame.

In addition to representing objects such as aircraft, sensors, ships, etc., the representation language must also depict a node's knowledge of events. In order to do this we have borrowed some ideas from the work of Schank and Abelson [14] and combined them with our frame based approach. In particular, an event is represented as a frame in which the descriptive attributes of the event (if there

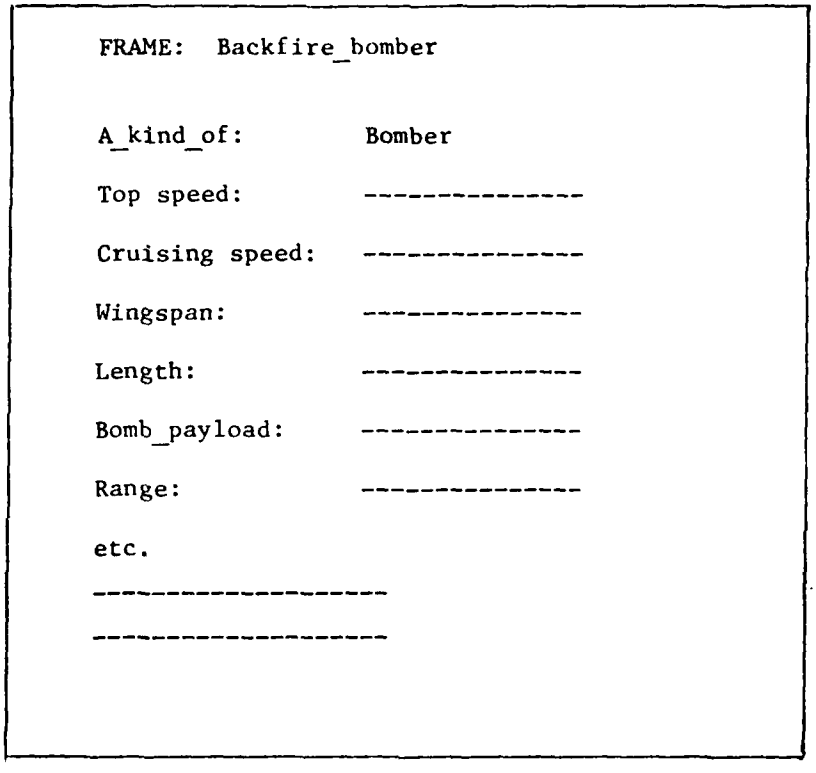


Figure 4-1: A Frame Representation Example

are any) are represented in the attribute value notation just discussed. In addition, the event frame also depicts a sort of "script" that portrays a set of simpler events that comprise the structure of this event. The sequencing relation among the component event such as sequential ordering, parallel occurence, conditional occurence, etc. are expressed in our representation system by use of a set of primitives called a "strategy representation language" (SRL).

Figure 4-2 is a simplified example of an event representation. Note that in the script for the event, exemplified reference is made to other event representations called "sensor_not_detecting" and "sensor_detecting_object". These two component events are themselves composites that are described in terms of simple events, and so on until a set of basic "primitive events" are reached in various event representations. This provides another kind of relationship among representations, namely that one event representation can provide a "high level" description of the flow of an event and yet the system can break this high level description down into a more detailed description by tracing the subevent descriptions. The notation "SEQ" in the figure is the relational operator from our SRL which represents that the component events occur one after the other. The constraint indicates that there is no overlap. Finally, we note that the representation of time is based on time intervals rather than point. The notation "NIL" in an interval representation means that a value need not be specified in an instance. (The notion of a time interval assumes that the first point does not follow the second.)

The approach to a representation of objects and events just sketched is a potentially quite powerful one. It borrows many of its ideas from earlier work in AI on knowledge representation with specific modifications and extensions that we have adopted to meet the needs of our distributed system applications.

There is yet another type of knowledge that must be represented in a node, and that is properly considered part of the node's static knowledge. We refer to the procedural knowledge held by that node. That is, in addition to knowledge in the form of descriptions of the objects and events that comprise the node's understanding of the task domain, the node must also have "skills". In the manner that is common to most work on expert systems, we represent these skills as knowledge in the form of situation-action entities we will call "knowledge

```
EVENT:  New_sensor_detection

A_kind_of:  Observation
Components:  Set_of:

                Observations_device:  Sensor
                Observor:  Node

Script:
  Subevents:  Get_of:
    Sub_event_1:
      description:  sensor_not_detecting
      time_interval  (NIL, T1);
    Sub_event_2:
      description:  sensor_detecting_object
      time_interval:  (T2, NIL)
  Script_program:
    Sub_event_1 SEQ Sub_event_2
  Constraints:
    T1 < T2
```

Figure 4-2: An Event Representation Example

sources" (KSs). More specifically, a KS is a representation of some action that is to be taken by the node as a response to the occurence of a particular condition, that we will call the "trigger condition" for the KS. There are a wide variety of ways to represent such situation-action entities. For now the reader may think of a KS as a kind of "if-then" rule in which the trigger condition is the "if part" of the rule and the action is specified by the "then part" of the rule. (We will modify this oversimplified presentation in a moment.)

Thus, a node's procedural knowledge, represented as these KSs, represents that node's ability to take actions in the domain and to solve problems or make decisions. The abilities of a DSN system node to participate in performing operations on the system's task will be modeled by the KSs held by that node. This includes the knowledge that thee node has for reacting to inputs from other system elements such as sensors, for using new and old information to make decisions. to combine information from various sources and pass a composite description along to yet other system elements, etc.

Figure 4-3 exemplifies more accurately the design of a KS. Note that this representation is also a frame description. However, there are certain attributes in this frame description whose value are so-called "attached proceures". In a KS, these attached procedures are executable code that embodies the actions that the KS will take when executed. Let us discuss a little more carefully the structure of the KS representation and how a KS is executed.

Recall from our earlier discussion that the basic notion a KS embodies is that of a "situation-action" rule. The KS represents some action that is to be undertaken under the condition that a particular condition or situation is true. The situation is represented by some pattern (i.e., frame description), or combination of patterns, in the system's knowledge base. The trigger of the KS describes this pattern. If the trigger pattern is ever asserted in the system's knowledge base, then the appropriate code is handed to the node's top level interpreter for execution. This code is the part of the "action" attribute called "code".

There is one more detail to discuss regarding the execution of a KS. There is an attribute in the KS frame called the "precondition" which affects whether or

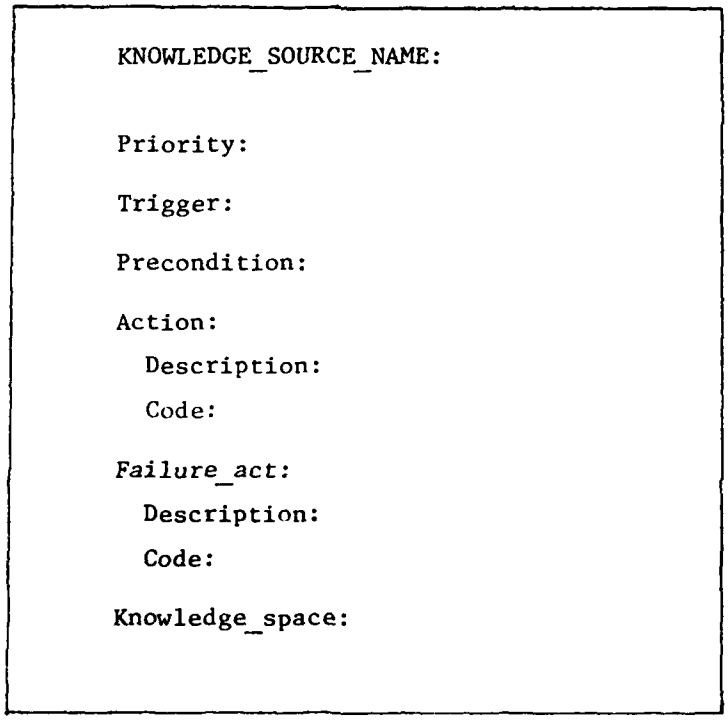


Figure 4-3: Structure for a Simple Knowledge Source

not a KS action is ever executed even though it has been triggered. Specifically, when a KS is ready to be executed by the top level interpreter, the precondition is checked. If it is not true then the code that is executed is that contained in the "failure_act" attribute. This code can represent a range of capabilitites such as test for data needed by the KS to perform its actions, checks to see whether other KS actions that are prerequisite to this KS's actions have been performed, specification of how to suspend the call to execute this KS action and (optionally) how and when to resume it later, etc. We have heretofore been informally describing the KS as a kind of "if-then" rule. With the addition of the precondition and the precondition failure_act the more proper description of a KS is as an "if-then-else" type of rule.

There are some additonal features of a KS that need to be discussed. First, both the "action" and the "failure_act" attributes of a KS contain another entry besides the code. This entry is called a "description". It is important to understand that the code entry for an "action" or "failure_act" is intended to be executable code. In order for the node to "know" about what a particular KS does, some symbolic description must be provided that can be interpreted. For example, *suppose that the node has established a goal to accomplish some action.* Then that node must be able to examine its KSs to see if the actions described by any of them accomplish the desired goal. Some goal seeking KS must read the descriptions of other KSs until one (or some combination) is found that describe actions that achieve the goal. Suppose that some such KS is found whose action description matches the goal. Then the next step in goal seeking is to try to ensure that the trigger and precondition patterns for that KS are asserted. Therefore, these trigger and precondition patterns now become subgoals; if they can be asserted then a chain of actions that lead to the goal can be executed. This exemplifies how goal directed inference is achievable by a node due to the descriptive component of a KS.

Note next the slot called "knowledge_space" in the KS frame. This is intended to model the need that certain procedural elements have for saving information between executions. For example, suppose that a KS is intended to be a counter which keeps track of the number of occurences of some event and reports when some number of occurences has taken place. One way to represent this is with a KS that is triggered by occurences of that event and which

increments a local datum and then "goes to sleep" until triggered again. When the KS internal count reaches the requisite number on some activation the report will be made.

The local data space attribute of a KS thus allows it to keep local state information. Actually, the design we have chosen also provides for separate KSs to share portions of their knowledge_space with each other (but without making this data globally accessible). This provides a fundamental way that procedural elements can communicate state information with each ther. There are other advantages to allowing a KS to have a local data space. However, we will defer discussing these for a few moments.

The final KS attribute to mention is the "priority". This is intended to be a very low resolution representation of the importance that this KS has when it is triggered in comparison with the other KSs in the system. As we shall see when we discuss the top level control structure for a node, this priority is really intended to be used as only a very rough estimate of urgency. The actual way that the node orders triggered KSs for execution relies on other mechanisms that can operate in a more context specific way.

We have not yet mentioned one very important aspect of the KS representation used in our system. We have restricted the discussion of a KS action to being a single piece of code that is directly executed when the KS is triggered. It is also possible for a KS to be a representation of a composite action by making use of the local knowledge_space in a KS and the script language we discussed in our consideration of event representations. In the case of a composite KS, the code (for either the action or the failure_act) can be a simple action that executes an entire group of actions in some order. The representation of these actions is located in the KS's knowledge_space. These actions are organized in a control structure in terms of the strategy representation language (SRL) discussed earlier. Thus, the code acts like a local interpreter of a script for composite action.

Let us summarize the basic features of a KS as a model of procedural knowledge. It is an if-then-else construct which provides for important capabilities such as synchronization, type checking, etc. to be expersed. It provides for event directed execution by its trigger-action basic structure, and for goal

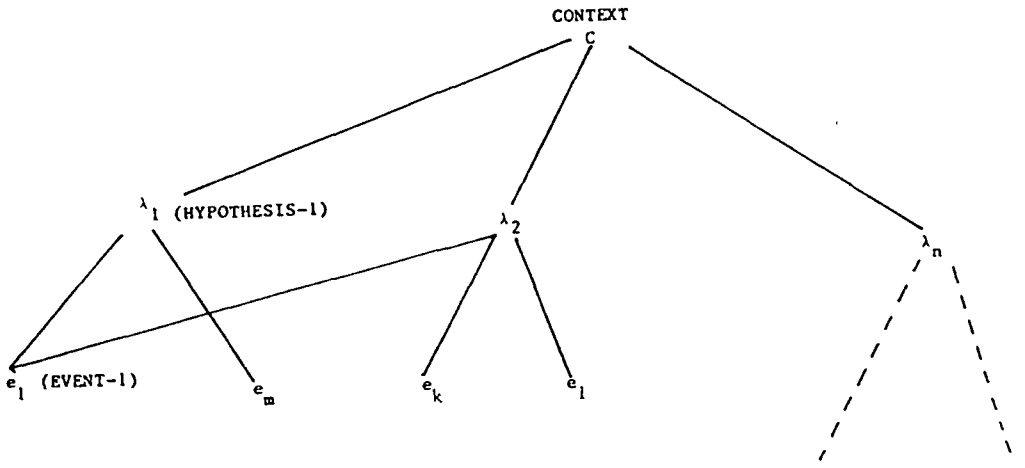
directed execution by having descriptions of the outcome of its execution. This type of model for a procedural element in a system gives generality to the kinds of inference that a system can do using this KS structure as the format for expressing its procedural knowledge. Finally, the KS provides for local data storage so that both computational and data state can be saved between executions. One very important way that this local data space construct is used is in the representation of composite KS actions. In this case the script for a composite action is found in the KS's knowledge_space and the code is an interpreter that executes this script. Thus, the KS model we have chosen is also sufficiently general to represent very powerful computational constructs such as object-oriented code [15].

We have now discussed the basic concepts used in the design of a node for representing static knowledge. Let us summarize this discussion. A node's static knowledge is represented in a frame description language that portrays a concept as a structured representation of attributes and the values or ranges of values that each attribute may have in any instance of the concept. In addition, events are represented as frames which contain a script depicting the ordering of simpler events in addition to the descriptive attribtes of the event representation. Finally, KSs are represented as frames with attached code that may be executed to enact the procedural knowledge that the KS represents. KSs may be simple, meaning that the action to be carried out is one uninterruptable activity, or they may be composite. In a composite KS the code is a (usually trivial) interpreter that executes a set of actions described in the KS's local knowledge_space by a script.

4.2.1.2 Situational knowledge

Now we should consider the node's representation of situational knowledge. This type of knowledge refers to the node's hypotheses regarding the existence of objects, events, constraints, goals to be met, etc. in the task context. One technique we have devised for representing such hypotheses is depicted in Figure 4-4.

In that figure one may see that each hypothesis is represented as some pattern of underlying assertions or observations. One may think of this as a



EXAMPLE HYPOTHESIS

λ_1: 2 TARGETS DETECTED

EXAMPLE EVENTS

e_1: OBSERVATIONS IN SET A FROM SAME TARGET

e_2: OBSERVATIONS IN SET B FROM SAME TARGET

Figure 4-4: Format for Hypothesis Representation

represetation in which the hypothesis is a claim that some compound event has occured (or is occuring), and the underlying observatons are the evidence that supports the hypothesis. The individual pieces of evidence may simultaneously support more than one hyothesis. Each piece of evidence can have some value attached to it that represents the plausibility of that piece of evidence, and the association of a piece of evidence to the hypothesis is weighted. The plausibility of the overall hypothesis is a function of the plausibilities of the supporting evidence modulo the weight of association for each piece. Finally, hypotheses are organized in clusters that represent mutually competing hypotheses for interpreting a set of observations or other assertions. In this way groups of hypotheses that represent conjectures about unrelated sets of data are partitioned from each other so as to simplify the process of hypothesis management. The techniques of hypothesis representation and management have been discussed in the report [1].

Figure 4-5 outlines the essentia' notions that enter into the process of hypothesis management. For the most part the notions presented there are quite obvious. However, there are actually a number of alternative methods for accomplishing many of these functions. For example, the task of propagating the plausibility of supporting evidence to the plausibility of the hypothesis that it supports may be handled in a variety of ways (e.g., using a Bayesian approach, Dempster-Schaffer theory, fuzzy logic, etc.). In turn, the pruning rule for eliminating hypostheses when their plausibility falls too low may be one of a large number of alternatives. We have not yet determined which, if any, of the alternative computational theories for combining evidence is the best; however, the representational calculus exemplified in figure 4-4 provides a structure within which any of these alternatives can be applied.

4.2.1.3 Planning knowledge

The next issue to be discussed in the representation of a DSN system node's knowledge has to do with the potential need for a node to engage in planning, such as resource allocation, adaption, etc. Although it will not be a requirement for all nodes to have planning ability, we nevertheless must provide this capapbillity if we are to use one general model of a DSN system element as a template with which to model the various individual system elements, some of which will

HYPOTHESIS MANAGEMENT

- PRUNING
  - — ELIMINATING LOW CONFIDENCE HYPOTHESES
  - — SETTING ADAPTIVE THRESHOLDS
- COMBINING
  - — REPLACING SIMILAR HYPOTHESES BY ONE HYPOTHESIS
- CLUSTERING
  - — PARTITIONING HYPOTHESIS SPACE
    - - NO POSSIBLE CROSS ASSOCIATIONS
- RECOVERING
  - — ELIMINATE HYPOTHESES GENERATED BY NOISE

Figure 4-5: Hypothesis Management Actions

indeed be capable of planning.

Actually, the planning capability of a node is representable in terms of three constructs that we have already introduced. The first of these is the representation of events, which a node may use to represent goals that are to be achieved. The second is the SRL used to describe the ordering of activity in composite events and in composite actions. These two devices provide the means to a node to represent plans for a series of actions to achieve goals. The third mechanism is to provide a node with a specific set of "planning" KSs. These KSs will include KSs to post goals to be achieved, KSs to divide goals into groups of subgoals, KSs to construct action scripts to achieve goals or subgoals, KSs to execute a plan once constructed, KSs to monitor plan execution so as to provide for replanning capability when conditions that were part of the assumptions of a plan have changed, etc.

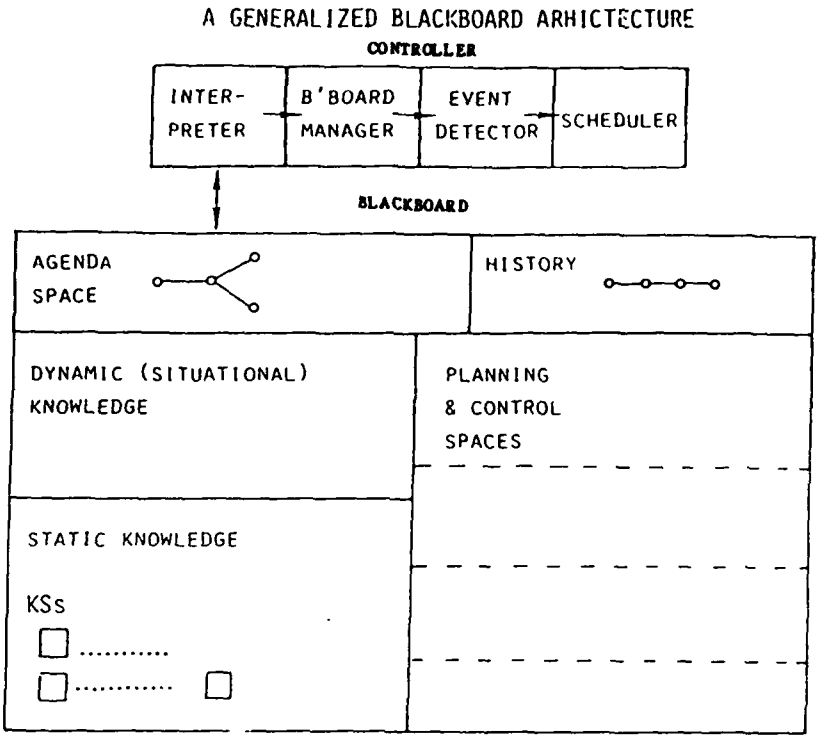
4.2.1.4 Control knowledge

The last major type of knowledge that we promised to discuss is so-called control knowledge. This refers to the knowledge that the node has for how to organize the actions that it must undertake. If we are to model nodes in a DSN system that are intelligent we must provide for a control structure that is flexible and context sensitive. Thus, in addition to providing KSs in a node model that can carry out actions in the task domain, we also need to provide the node with KSs that can resolve conflicts if more than one KS is relevant to a situation and KSs that are able to alter the queue of pending actions that that node must execute in response to a sudden change of conditions. The approach we take to this is to apply the planning capability just discussed to the control of the node own computational processes. Thus, our approach to control is to view it as a planning and problem solving domain for which we supply a set of control KSs that are triggered by the structure of actions that are pending and that are ongoing in that node. This approach is an alternative to providing a single complex control structure at the top level in a node.

Of course, the topic of control raises the issue of the general framework for top level control that is to be provided for a DSN node representation. Figure 4-6 depicts this framework. The general structure of the architecture is divided into the two components of the CONTROLLER and the BLACKBOARD. This architecture is an intellectual descendant of the type of expert system known as a blackboard architecture [16]. In representing a DSN node in this way, the entirety of the node's knowledge is stored on that node's blackboard. As is shown in the figure, this includes an area for the static knowledge held by the node as well as an area for the node's hypotheses. Specifically, the blackboard contains the representations of the node's KSs that represent the actions that that node is able to perform. Finally, there is an area (shown at the top of the blackboard) that represents the node's notation about its current state of execution, called the AGENDA SPACE. In this latter area is a representation, called the AGENDA, of the set of KSs that have already been triggered but have not yet been executed by the CONTROLLER. There is also in this space a HISTORY structure which is a kind of audit trail of KSs executions that have already occured.

The other major component of the framework is the CONTROLLER. This is the algorithm that controls the general execution of the node model. It is divided into four stages. These are the INTERPRETER, the BLACKBOARD MANAGER, the EVENT DETECTOR, and the PRIORITIZER. The basic loop repeated by this CONTROLLER is essentially the following.

First, it is the responsibility of the INTERPRETER to select the first KS on the agenda, check the KS precondition, and execute the action or the failure action for that KS. Then the BLACKBOARD MANAGER filters the output of the code run by the INTERPRETER and posts any changes to the BLACKBOARD that have been created. Next the EVENT DETECTOR checks the contents of the blackboard to see if there is a pattern of assertions which match the trigger pattern for any KS. The list of KSs that are triggered are passed up to the PRIORITIZER which inserts the newly triggered KSs into the remaining AGENDA, ordering them on the basis of their priority. The KS that was at the front of the list of the AGENDA is removed and is added to the HISTORY_LIST. Then the cycle is repeated again.



- MIXED INITIATIVE PLANNING/REPLANNING SUPPORTED
  - — HIERARCHICAL PLANNING CAPABILITY
  - — OPPORTUNISTIC PLANNING/REPLANNING POSSIBLE
- COMBINES ANALYTIC (E.G. BAYESIAN) AND HEURISTIC APPROACHES TO EVIDENTIAL REASONING
- REAL TIME APPLICATIONS FEASIBLE
  - — SPEED DUE TO COMPILED FORM OF PROCEDURAL KNOWLEDGE PLUS A FAST TRIGGERING MECHANISM
  - — AGILITY DUE TO USE OF CONTROL KNOWLEDGE SOURCES TO SHIFT ACTIVITY FROM ONE PATH TO ANOTHER
- OVERALL ARCHITECTURE PROVIDES A SYSTEM DEVELOPMENT ENVIRONMENT FOR RAPID CONSTRUCTION AND PROTOYPING

Figure 4-6: Architecture for a DSN Node

There are a number of efficiencies that we have included in the actual design just sketched. First, note that the way the INTERPRETER executes a KS action requires only that the code for that action be passed off to the underlying cpu of the system in which this system is implemented. That is, when a KS code is executed the INTERPRETER is not performing translation from a symbolic sturcture to an executable one. The executable structure is already present in the code attribute of the KS, and the INTERPRETER merely passes this executable structure on to the underlying system as a process. In addition, the step of determining what new KSs have been triggered is embodied in an algorithm that we have developed that involves almost no search. Thus, triggering is very fast (and in fact, our algorithm provides for triggering on deletions as well as additions to the blackboard and for use of full generalizations in the trigger patterns that may be used).

In the discussion of knowledge representation above we pointed out how a composite KS can represent a complex algorithm. Coupled with the top level architecture just sketched, this provides an architectural template that can be used to model a simple fixed control system as easily as we may use this approach to model a complex, intelligent node. A simple node can be represented by a model with only one KS in which the code for that KS is the fixed algorithm that the node performs. (Note that there is almost no overhead to pay in using the CONTROLLER to run that KS.) The blackboard for that simple node would probably contain very little additional knowledge. On the other hand, a complex intelligent agent could be modeled by a version of this template containing knowledge representations of all the types discussed above, including a number of simple and composite KSs representing the decision making and problem solving abilities of that intelligent node. We have used this facility in a Schemer application in which we constructed a distributed computation system with distributed performance management and performance diagnosis abilities [10].

This completes the discussion of the features provided in our testbed design that are to be used for modeling a participant, a node, in a DSN system. The DSN system designer is provided with techniques for modeling the capabilities of even intelligent participants in a DSN system by being provided with the expert system framework depicted in figure 4-6, plus the representation language for

describing that node's knowledge of objects, events, rules and other procedures, as well as hypotheses about the task context and techniques (i.e., KSs) for managing these hypotheses. We again should note that, although the modeling facilities just described are especially intended for the modeling of intelligent DSN system elements, these facilities are quite well suited for modeling simpler elements of a DSN system. In particular, the minimal amount of structure provided in the top level controller for a node model minimizes the computational overhead, and the computation modeling assumptions, that are implicit in the use of the exert system framework just described as a model of a DSN system element.

4.2.2 Node relations in the DSN system

Now we must consider how the designer of the DSN system is to model the relationships that obtain among the elements of the DSN system and the operational context within which the system is to perform its functions. The basic concept for how we propose to accomplish this is really quite simple. First, we need to provide a modeling framwork within which the models of the various nodes, constructed as discussed above, can be embedded. It is within this larger modeling context that the relations among nodes and the characteristics of the task domain are to be modeled. Second, within this larger modeling context the relations among the system nodes and the objects and events of the task domain will be represented.

Let us review the critical relations among system nodes that must be modeled. There are two fundamental types of internode relations that must be represented in a DSN system design, the modeling of the communications relations among the DSN system nodes and the modeling of the control or authority relations that obtain among the system elements.

The modeling of communication relations among nodes involves the following basic notions. Suppose, for example, that we wish to model a specific communication between two nodes, say Node_A and Node_B. Each of these nodes is, of course, represented by a structure like that of figure 4-6. A communication from Node_A to Node_B can be modeled as an output of some action of the first

node that is represented in the larger modeling framework in which the two node models are embedded. This communications output from the first node can serve as input to a structure that models the communications medium through which the message from Node_A travels. The message is processed by the communications medium model whose output is the input to Node_B's message receiving procedures.

The communications medium model will be constructed so as to model some specific mechanism for dealing with a particular type of message transmission such as, for example, a packet radio network. The model of such a communications medium can be constructed using the same approach as that used to model a system node. This model of a communications medium will contain procedural elements (KSs) that represent operations of the communication medium in receiving a message, processing it, and then passing it on. In addition, the communications medium model can also include KSs and other elements that represent the effects of noise or other perturbations on the message passing process. Finally, the communications medium model passes on the results of its processing to the receiving node by producing output in a form that represents proper input for the message receiving KSs of the intended recipient node. (This is, of course, only true provided that the message has not been "blocked" in its transmission as a result of processes of noise or other interference being modeled).

This approach to modeling the communications that occur among DSN system elements requires no general modeling facilities beyond those already provided for modeling the decision making elements of the DSN system. It has the advantage of providing the system designer a way to independently model the four primary components of internode communications. These are; message composition and sending, message transmission, transmission interference, and message reception and interpretation. Using this approach it is possible to model in great detail a variety of communications methods being simultaneously used in a DSN system.

The other fundamental internode relation to discuss here is that of control or authority relations among system nodes. The modeling techniques just described for depicting communications relations can be extended slightly to handle this modeling chore. In other words, control by one node over the actions

of another may be modeled as a type of communication plus "rules" held by each node regarding the bounds of authority. In particular, a "command" from one node to another can be a communication that, when processed as input by the second node, is the trigger condition for some specific action (KS) of the recipient node.

4.2.3 Task context modeling and simulation

Let us now consider how we may use the same modeling approach just described to model the task environment in which the DSN system will operate. By now the reader should be able to anticipate the gist of this discussion. The functional elements of the task domain are modelable by the same means that are used to model the DSN system nodes. That is, each functional element is a modeled as a structure that produces output representing the effects of that element on the DSN system and on other elements of the task context. For example, suppose that the DSN system is engaged in a task of using sensors to watch the activity of vehicles along a road network. Consider a model of a specific object that is "moving" along a particular road. This object movement can be represented by a computational element that produces as output an update of its position over time. This output can be used as input by a particular sensor model that is constructed so as to be able to "observe" objects in an area occupied by the vehicle (model). This input to the sensor can, in turn, cause that model to produce an output (say, in the form of a commmunication) that is taken as input by one or more nodes which process information from this particular sensor. Thus, the model of the vehicle has caused a chain of activity that culminates in some set of decision making nodes of the DSN system in being called to respond to that domain object's activity.

In general then, each functional object in the task environment can be modeled by an instantiation of the architectural framework of Figure 4-6 containing data and other procedures specific to the functional properties of that object. The actions of this object produce outputs that are suitable inputs to specific DSN system components that are intended to "perceive" and respond to that object's activity. This modeling of cause and effect is very similar to the way in which communication relations among DSN system nodes are to be modeled.

The general testbed enviroment will, therefore, be required to provide a context for representation in which the specific models of the DSN system nodes, the communications media, and the task environment's functional entities can all be modeled. The activities of all these models are recorded in the testbed's data base as outputs such that each output can be discerned by all and only those models that are intended to be able to react to the activity represented by that output. The testbed data base holding the representations of all these actions of system and context elements, must be organized so that the output from any particular element will qualify as an input for only those other elements that are appropriate.

4.3 AN ARCHITECURE FOR THE DESIGN TESTBED

In the preceding discussion we have presented some ideas on how to model the elements of a DSN system. The modeling approach we have offered is to use a generalized expert system architecture as the basic "template" for designing a node in a DSN system. In addition, we have suggested that this same set of techniques can be used for representing the functonal elements of the task domain as well as for depicting the command and communication relations that obtain among the DSN nodes. We have not yet dealt with how a design environment might be provided as the framework within which to do this design construction, rapid prototyping, and evaluation.

We will now present an architecture that provides a context for the development of the types of structures just discussed along with facilities for using these structures to simulate the performance of the DSN system they are intended to model. Since the approach we have proposed relies upon an implicit analogy between expert system construction and DSN system design, we will offer as our general solution a framework that is essentially an expanded version of an expert system building environment. We present the design of an AI architecture for expert system construction called Schemer as the basis of a testbed environment in which to construct, evaluate, and run simulations of a DSN design. The basic ideas in Schemer are ones that we have had a great deal of experience with, having used a simpler version of this system than that presented here as an expert system development environment for construction of a number of expert

system applications. However, in order to deal with the problem of modeling a distributed system, and a system in which many of the elements are themselves intelligent decision making agents, we have found it necessary to greatly extend our original ideas.

The basic idea behind Schemer is to provide a computational environment in which an expert system designer can incrementally build his application by specifying the components of his system including the representation of objects that must be known to the expert system, and the procedural rules that define the system's capabilities for taking actions to carry out its intended tasks. Thus, the expert system development framework must provide the developer with a knowledge representation language to encode the expert system's knowledge of essential objects and events in the task domain and a general format for expressing the procedural elements of the expert system, its knowledge sources.

To provide this representational support Schemer is equipped with a general frame representation language for expressing such knowledge, and a special format for representing procedural knowledge. Not surprisingly, the approach we have chosen for knowledge representation has already been discussed in the previous section. The representation language approach discussed there for representing the knowledge of a DSN system node is precisely the type of language that we have used in Schemer, and the KS definition of Figure 4-3 is exactly the format used for representing the procedural elements of an expert system within the Schemer development environment.

In addition to constructing the elements of his expert system using the representational facilities just described, the developer must be able to run early versions of his system on test problems to determine whether the design to that point is achieving expected behavior. Similar to the ideas in the previous section we found that the basic representation facilities for constructing the expert system itself could also be profitably used to build a kind of simulation environment (i.e., a set of test cases) against which various versions of the system design could be tested. In fact, our approach to providing a simulation based, rapid prototyping environment is motivated by the experience that we have had in using the modeling capabilities of our expert system development environment to construct models of task situations against which that system could be tested. This also

means, of course, that the combined expert system model and task domain model would need to be able to be put in a form that could be executed. Hence, Schemer was required to be a context for controlling the execution of such design prototypes in the context of specific models of the task domain.

Figure 4-7 depicts the way that we designed Schemer to provide this support for both expert system design and prototyping. Essentially, the idea we pursued was to take the basic blackboard architecture for representing an expert system such as the type shown in Figure 4-6, and then to add some additonal levels of control to the architecture to support design construction, scenario context construction, and control of execution of the expert system on the model of the task domain. The structure of the blackboard in the figure is divided (for purposes of presentation) into a left half on which various levels of knowledge representations are written, and right half on which the procedural KSs are stored. Vertically we have depicted four basic levels of representation. At the lowest level there are the knowledge representation structures and the KSs of the expert system model itself. This level is the part constructed by the developer that will eventually stand alone as the expert system application. Above this level is the representation of the simulation environment for testing the expert system design. For the most part the task model is represented by KSs that provide simulated events as input to the expert system KSs. These task domin KSs do this by posting their output on the part of the board labeled "Task Domain Event Reps." These event representations serve as triggers for the expert system KSs at the level below. This approach provides a way of simulating the interaction between the expert system and events in the task domain.

The upper two levels of the blackboard in figure 4-7 represent the way we have provided automated tools in the original Schemer development environment. The third level from the bottom represents the presence of some evaluation tools, encoded as KSs, and the representation of the results of evaluation on the left. The level immediately above this is the set of KSs that can be used to control the use of the system for various activities. For example, control KSs at this level can take the elements of a scenario and cause the domain model to post events of activity that, in turn, trigger the actions of the expert system in response. Also, we have used control KSs at this level to trigger special analyses to be performed on the expert system's activities upon detection of a particular event that has

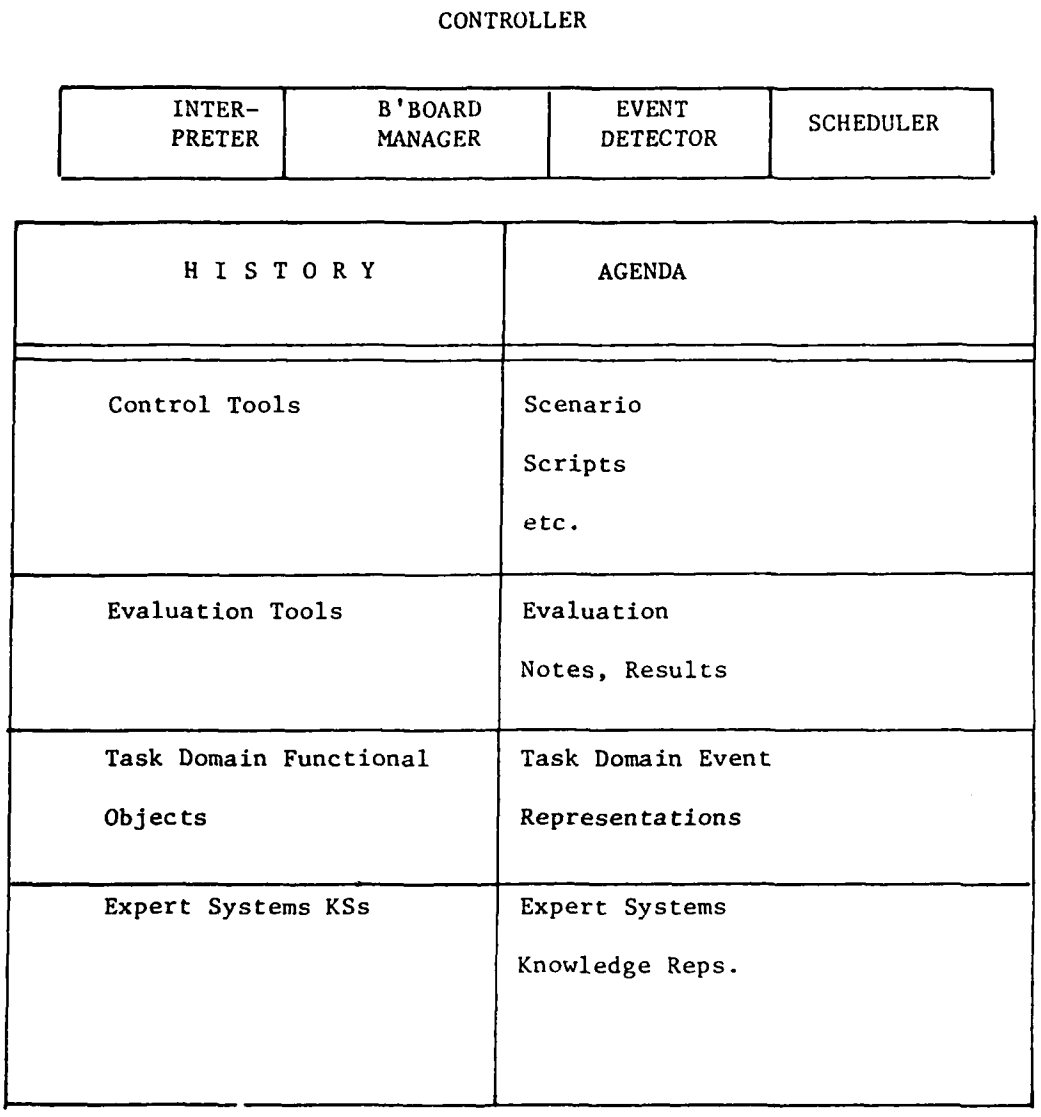


Figure 4-7: Basic Schemer Architecture for DSN Design

occured during execution of the simulation. Other forms of these control KSs that we have used include a KS that effectively "single steps" the execution of the expert system being evaluated and control KSs for displaying the data structures that "cause" triggering of the expert system KSs as these occur.

The top level control structure of the Schemer architecture is essentially the same as that we have described in the previous section. User interaction with the development environment is accomplished by having user post input events on the control blackboard which trigger appropriate actions by Schemer. Thus, this system can be operated interactively as well as in the fully automated manner that is assumed for this general type of architecture.

There remain a number of limitations to the current version of Schemer as a DSN development environment. First, little or no work has been done on the development of explicit evaluation tools beyond those tools to support the empirical observation and summarization of the expert system's performance on specific scenarios. Second, nothing is provided in the way of "library" facilities that contain frequently used control and structures that can be employed for a specific application [17]. Third, and perhaps most important, the expert system procedural components that can be modeled in the original version of Schemer are either simple condition-action rules or composite rules in which the action part is some algorithm for combining simpler actions. There is no provision made in this earlier version of Schemer for allowing a KS to be an entire expert system architecture of its own as we discussed in the previous section. Finally, we have yet to integrate non-AI algorithms and systems such as GTC into a particular Schemer application.

In Section 4.2 we discussed a set of representation and modeling techniques that we believe explicitly address the requirements of a DSN design testbed. In addition to these facilities we are condsidering some additional extensions to the previous version of the Schemer architecture. Along with the analysis of representation from section 4.2 we have devised the changes depicted in Figures 4-8 and 4-9 to address the limitations of the original Schemer development environment for use as a DSN system testbed.

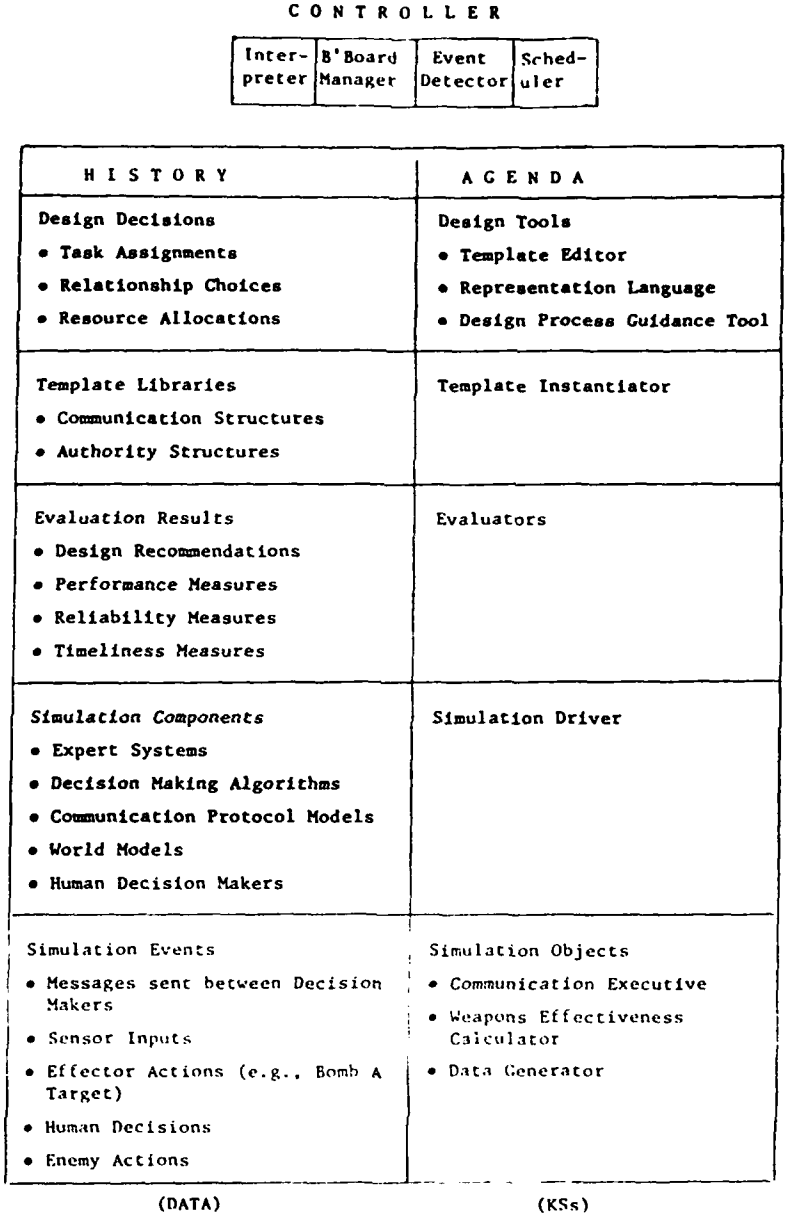


Figure 4-8: Full DSN Testbed Architecture

In Figure 4-8 we see five levels of the testbed blackboard. The lowest level corresponds to the modeling of the DSN system and the task context of that system. The level above that is used for knowledge about the state of the simulation (on the left of the diagram) and KSs that are used to control the execution of a simulation from a script or other description of that scenario provided in the space to the left. These areas generally correspond to facilities that were also available in the earlier Schemer design, although the system facilities for simulation control are now separate from those for design. The third level from the bottom of the figure refers to tools for, and knowledge about, evaluation of a test of a design on a scenario. In other sections of this report we have discussed techniques for evaluating some portion of a design analytically. Certainly these techniques should be incorporated as evaluation tools that can be called to use by the DSN system developer for analysing or verifying the construction of his system. However, the complexity of a DSN system precludes the use of such analytic techniques in many cases. Therefore, another approach is called for. There are a number of statistical and other techniques that can be applied to the evaluation of a system's performance by analyzing that system's performance data and comparing it to some standard. For example, suppose the expert system being developed is intended to construct hypotheses that explain the sensor data. A set of evaluation tools that might be provided could be some to compare the assertions being hypothesized by the expert system with the "ground truth" represented in the scenario description that is driving the simulation of that expert system's task performance. We believe that this simulation based approach to a design methodology can fruitfully employ a great number of such statistical tools since these tools yield useful descriptions of such things as the effectiveness or accuracy of system behavior.

The next level up in the figure represents the presence of libraries that can provide templates of types of structures that are likely to be useful in a wide variety of design situations. This can include templates that describe commonly used forms of DSN system components, templates for specific types of communications relations among nodes, or command structure templates. Additionally, it would be extremely useful to have a library of standard test cases that could be used by system designers developing DSN systems for well known classes of application. The testbed could provide assistance in the form of template "instantiators" that can be called into use by the designer to insert a specialized form of

some template into his design.

Finally, the next level of the system provides for a set of tools to explicitly aid the designer of a DSN system in both the construction and test of his design. We have done some experimenting with this notion in our work on Schemer by looking at ways to simplify and automate the more repetitive aspects of the system designer's task. For example, we have developed a format for constructing new KSs in the system which structures the organization of the KS definition for the designer by providing a template for KS definition. When filled out this template is automatically used by a design tool to construct the actual KS and its underlying code object. Other tool we are experimenting with include checkpointing facilities and tracing facilities for examining the conditions that lead to a particular action by the expert system. An important aspect of this topic of design support is that of explanation [18]. Tools to provide natural language descriptions of the performance of a complex system like a DSN system are sorely needed. As a starting point existing AI techniques for producing explanations of system performance would be an important addition to the design testbed.

There is one more limitation of the earlier version of Schemer that we have yet to address. This is the need for modeling elements in a DSN system that have the complexity and sophistication of intelligent agents. In Section 4.2 we described the general model that is to be used to accomplish this modeling. This general model, shown in Figure 4-6, is itself a complete AI knowledge based expert system architecture. The problem then is to expand Schemer so that such a model can be embedded in a larger expert system framework. We have provided for this by expanding the definition of a knowledge source to that it can be instantiated as a complete blackboard subsystem with its own blackboard and its own internal KSs. Figure 4-9 exemplifies this type of KS. The example of Figure 4-9 indicates that the code of the KS can be a blackboard CONTROLLER and the data space slot of that KS can be an entire internal blackboard. We will refer to this type of KS as a "subsystem KS" (SSKS). This simple move provides an enormous increase in the modeling capabilities of the Schemer system. The overall blackbaord framework of the Schemer architecture itself can now contain embedded models of blackboard systems. These embedded expert system architectures can more fully model the complex intelligent agents in a DSN system. There are, however, two points that should be clarified in order to understand

SYSTEM ARCHITECTURE EXPERT

Priority: High

Trigger: User requests system architecture recommendations

Precondition: There are no KSs that determine implied architectural constraints awaiting execution in the agenda

Action:

Description: Read all architectural constraints on blackboard, elicit any required further inputs from user, try to find a system architecture to satisfy the requirements, post suggested architecture or reasons why architecture cannot be found to blackboard

Code: Interpreter

Inter-preter	B'Board Manager	Event Detector	Sched-uler

Failure_act:

Description: Move KSs that determine architectural constraints to front of agenda queue, then execute

Code: Interpreter

Inter-preter	B'Board Manager	Event Detector	Sched-uler

Knowledge_space: Constraints from blackboard, user inputs, architecture taxonomies

Figure 4-9: A Subsystem KS in the DSN Design System

how such a complex KS can be used in the encompassing blackboard framework.

First, the so-called "local" blackboard of an embedded SSKS can in fact be a shared area of an encompassing blackboard. That is, the content of the SSKS's blackboard can be a pointer to data actually external to that SSKS. This technique is needed to provide for the sharing of information that might be called for in some models. It could also presumably be a way of modeling how specific events that are external to an SSKS could trigger specific actions within that complex system.

The second point to be clarified has to do with the way that such an embedded architecture is controlled by the top level of Schemer. As with any other type of KS a SSKS is initially called into action by occurence of the event described in its TRIGGER slot. When actually executed the CONTROLLER (in the code slot of the SSKS) is used in place of the top level CONTROLLER. If the SSKS halts (i.e., its AGENDA becomes empty) then that KS execution can terminate just as simpler KSs do and the top level CONTROLLER resumes its operation. Under certain conditions, however, it may be necessary to be able to interrupt the operation of the SSKS in order to perform some special operation. An example of this would be the use of some evaluation KS that interrogates the knowledge base of the SSKS periodically to obtain, say, performance statistics on that SSKS. In order to provide for this type of control, the top level CONTROLLER can insert some annotations in the controller of the SSKS that causes that embedded interpreter to suspend periodically. When the SSKS suspends the top level can check for the occurence of assertions on the blackboard of the whole system that could trigger high priority KS to perform some action and, if appropriate, even cause the suspended SSKS to be aborted rather than resumed. This latter type of control would be needed if, for example, the designer had inserted a control KS that waited for the object being modeled by the SSKS to achieve some special state and then halt the execution of the simulation of which the SSKS was a part.

4.4 SUMMARY AND DISCUSSION

Let us summarize the discussion of the testbed arhitecture just presented. We have capitalized upon the analogy of expert systems to decision makers by proposing a design methodology that treats the design of a DSN system as identical to the problem of designing a system of distributed, cooperative expert systems. The architecture we have developed and implemented in a preliminary form provides a very powerful and general computational model within which to devise the broadest class of DSN system structures.

This approach to a design methodology is quite radical in its emphasis upon techniques of empirical observation of system behavior as a means of evaluating system design. Becuase of this emphasis on performance observation and analysis, a major requirement of the testbed is that it supports the construction of test scenarios that can be used to elicit such performance from the system. Thus, the ability of this testbed architecture to support rapid construction of such scenarios is almost as significant as the rapid development of the DSN design itself.

However, as central as we hold the empirical, rapid prototyping approach, we do not adhere to this approach to the exclusion of the use of other more analytic techniques of design evaluation. It is a strength of this approach that such techniques can be readily incorporated in the testbed as specific tools for evaluation that can be used in conjunction with the simulation and observational facilities of the testbed.

For the present project we are focusing on the development of the application of our testbed concept to the design of DSN systems that are built upon the use of the GTC and the information fusion algorithm as the fundamental node capability. As noted above, we have been fortunate in having been able to gain a great deal of experience with this concept by actually implementing partial version of such a testbed for use in the development of various expert system applications. However, the type of testbed architecture plus the full range of capabilities we have presented here has yet to be implemented. We see this additional effort as an important next step in assessing the validity of our concepts for DSN system development and for their value in assessing other technologies such as

the Generalized Tracker/Classifier (GTC) or other more analytic algorithms.

5. SIMULATION EXAMPLES

5.1 INTRODUCTION

A simulation of a distributed sensor network was performed to test and evaluate algorithms and to explore various technical issues experimentally. The current simulation focuses on the processing within each node and a perfect communication model is assumed. Each node in the network is equipped with a GTC (Generalized Tracker/Classifier) which processes the local sensor data and an information fusion module which fuses the information sent from the other nodes with the local information. The present status of the simulation effort is as follows:

- The communication pattern is arbitrary. Any communication between any two nodes can be set up.

- The maximum number of nodes in the network is four. There is no conceptual difficulty in increasing the limit on the number of nodes but simulation time will increase substantially since a single computer is used to simulate a distributed system.

- The processing of the local sensor data is by means of the GTC developed in the previous project [1]. Information fusion is based on algorithms for hypothesis formation and evaluation described in Section 2.

- The language for the simulation is Lisp. Lisp has been chosen because of the data structure and the plan of converting the simulation to run under the architecture of Section 4.

In order to handle arbitrary communication patterns among the nodes, the information fusion algorithm includes mechanisms to trace the histories of the tracks and hypotheses in the information graph. Without any loss of generality, information fusion from multiple nodes is carried out sequentially in a binary form, i.e, to fuse the information from node A, B and C, we first fuse that of A and B, and then the result is fused with the information from C. This simplifies the implementation of the fusion algorithm considerably.

In this section, some simulation results for two-node and four-node sensor networks are presented to illustrate the performance of the DSN fusion algorithm. For both examples, a simple discrete-state road network scenario was chosen and the target dynamics were assumed to be Markov with the road-segments as states. The main reason for using the simple target dynamics and scenario was to minimize any unnecessary numerical complexity due to target motion and to concentrate more on issues resulting from arbitrary communication pattern. The simulation program, however, is capable of handling more complicated scenarios if the appropriate algorithms are included.

The underlying models in the scenario are:

a. Targets are moving along the road network with discretized straight-line segments.

b. The target dynamics are Markov with a given transition matrix.

c. Each sensor measures position (segment number) along the road with some uncertainty due to the bearing and range measurement noise. Each sensor also has certain masked regions which it cannot observe.

d. The probability of detection of a target in each road-segment by a sensor is a function of sensor masking and the relative sensor location.

In addition to this, independent and identically distributed target models have also been assumed in the current simulation.

5.2 TWO-NODE COMMUNICATION NETWORK

First we consider the two-node case, assuming each node has only one sensor. The sensors observe the same road network although they have different fields-of-view. The road network and the location of the sensors are shown in Figure 5-1.

5.2.1 Scenario

Each individual target position is represented by the segment number and its evolution is assumed to be a Markov process. The target state at any time is thus characterized by a probability distribution on the road segment. Because of the terrain, the two sensors have different masked regions (see Figures 5-2 and 5-3). Each sensor generates a measurement in the following way. The detection of a target at state x_i by a sensor depends on the detection probability which is 0 whenever the target is in a masked region relative to the sensor. For any detected target located at x, the measurement y, which is also a segment number, is generated according to the following conditional probability distribution function: (see Figure 5-4)

$$p(y \mid x) \approx \sum_i \alpha_i(y_i) U(y_i) \tag{5.1}$$

where

$$\alpha_i(y_i) = \int_{r_{min}(y_i)\theta_{min}(y_i)}^{r_{max}(y_i)\theta_{max}(y_i)} \int g_r(r \mid \bar{r}(x)) g_\theta(\theta \mid \bar{\theta}(x)) dr d\theta \tag{5.2}$$

$U(y_i)$ is a uniform function on segment y_i with unity value and $g_r(r \mid \bar{r}(x))$ and $g_\theta(\theta \mid \bar{\theta}(x))$ are sensor characteristics corresponding to the the measurement uncertainty in range and bearing given the average range and bearing of a particular target location x. False alarms are also added according to the sensor model.

The total number of targets is constant but unknown and its a priori distribution is Poisson with mean ν_0. The number of false alarms in each scan is also Poisson with mean ν_{FA} for each sensor. The target positions are independent and identically distributed with the a priori distribution uniform over the road

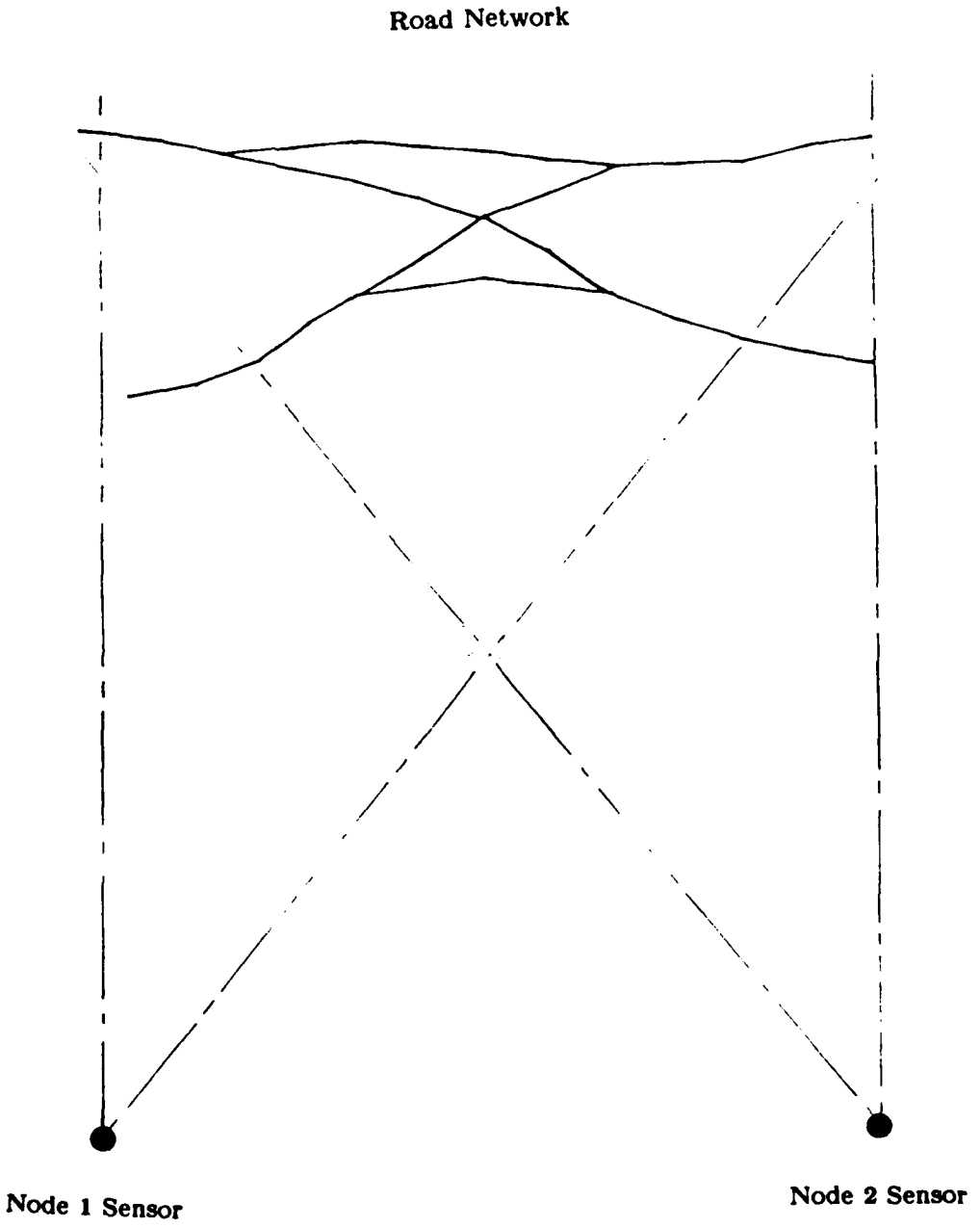


Figure 5-1: A Two-node Scenario

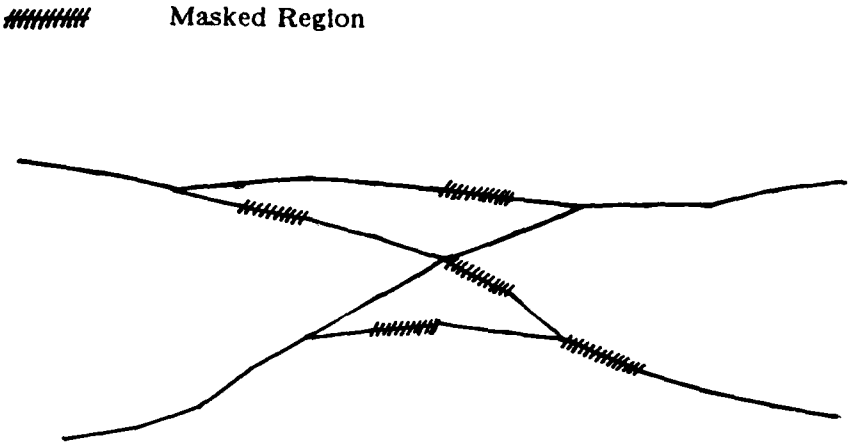


Figure 5-2: Masked Regions of Sensor 1

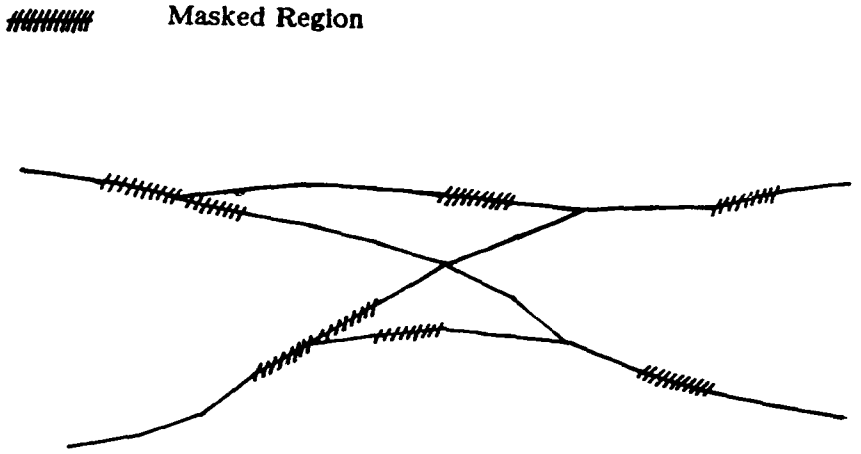


Figure 5-3: Masked Regions of Sensor 2

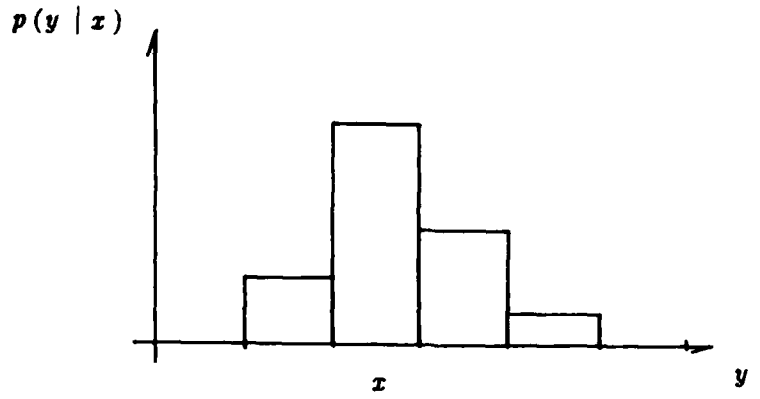


Figure 5-4: Conditional Probability Distribution $p(y|x)$

network states, and targets are expected to move into the field-of-view from the edges at any time.

5.2.2 Communication Schemes

Three types of communication schemes were examined:

a. Broadcast communication: the two nodes share and fuse their information every third scan.

b. One-way communication: node 1 sends information to node 2 every third scan but node 1 gets nothing from node 2

c. Decentralized: the nodes do not share any information

If sensor data are broadcasted every third scan, then the broadcast case can be regarded as a centralized scheme where the central node receives all the measurements periodically. In the case when only tracks and hypotheses are communicated, the processing is essentially hierarchical with intermediate results available at each node between periodic instants. The decentralized case is the other extreme situation. In each case, all the hypotheses at each node are communicated. The parameters used in the simulations are given in table 5-1.

In each simulation, all the hypotheses were examined and compared to the true trajectories of targets according to the measurement-to-target association histories. The hypothesis best matched to the ground truth is defined as a true hypothesis. The most likely hypothesis (highest probability) is called the best hypothesis.

Expected number of targets		ν_0	4
Expected number of false alarm		ν_{FA}	1/scan
Probability of detection		P_D max	0.9
Measurement error	range	σ_r	0.5 (km)
	bearing	σ_θ	0.2 (radius)
	radial velocity	σ_ρ	0.1 (km/min)
Pruning threshold		ϵ	0.05

Table 5-1 Simulation Parameters

5.2.3 Simulation Results

In the following we present the results of some sample runs. More extensive simulation results will be performed when the development of the test bed is completed.

The target scenario (ground truth) and the probabilities of detection are given in Figures 5-5 ,5-6 and 5-7. As shown in Figure 5-5, two targets A and B move from left to right on the road network, where the subscript on each target represents the corresponding time index. Note that in Figures 5-6 and 5-7, the detection probabilities are lower on the roads which are almost orthogonal to the sensors' lines of sight. This can be used to approximate the effect of the velocity Doppler on a MTI radar.

Figures 5-8 to 5-13 show the probabilities of the best hypothesis versus the true hypothesis at each scan for two nodes under different communication schemes. The results of the decentralized case are shown in Figure 5-12 and Figure 5-13 for node 1 and node 2. As can be seen from the figures, the performance of each node is quite poor. The probability of the true hypothesis only approaches that of the best hypothesis during the last four scans of node 1. For node 2, the probability of true hypothesis is never higher than 0.2 as can be seen in Figure 5-13. This is so because in the decentralized case, each node only processes its own local observations which are quite sparse with the given masked regions.

Figures 5-10 and Figures 5-11 present the results with one-way communication. Obviously, node 2 has better performance than node 1 reflecting the fact that node 2 receives more information than node 1 due to the communication pattern. Note here that the behavior at node 1 is exactly the same as in the decentralized case (see Figure 5-10 and 5-12). This is obvious because as in the decentralized case, node 1 does not receive any information from node 2. Similarly, during the first two scans, the results of node 2 are the same as in the decentralized case since no information has been received yet (see Figure 5-11 and 5-13). From scan 3 on, because the information from node 1 begins to arrive via communication, the probability of the true hypothesis increases and reaches 0.8 in a few scans.

The results of the distributed case given in Figure 5-8 and 5-9 for node 1 and node 2 clearly show its superior performance. Almost all the best hypotheses are the true hypotheses as can be seen in Figure 5-8 and 5-9. Comparing Figure 5-8 and 5-9 we can see that node 1 and node 2 have the same performance at the communication times, this is true because both nodes share the exactly same information when broadcast communication takes place.

5.3 FOUR-NODE COMMUNICATION NETWORK

There are now four nodes each with only one sensor observing the same road network.

5.3.1 Scenario

The sensor locations and their fields-of-view with respect to the road-network are shown in Figure 5-14. The target scenario is the same as in the two-node case. Because of the terrain, the masked regions for the sensors are different. The probability of detection for each sensor is shown in Figure 5-15 to Figure 5-18.

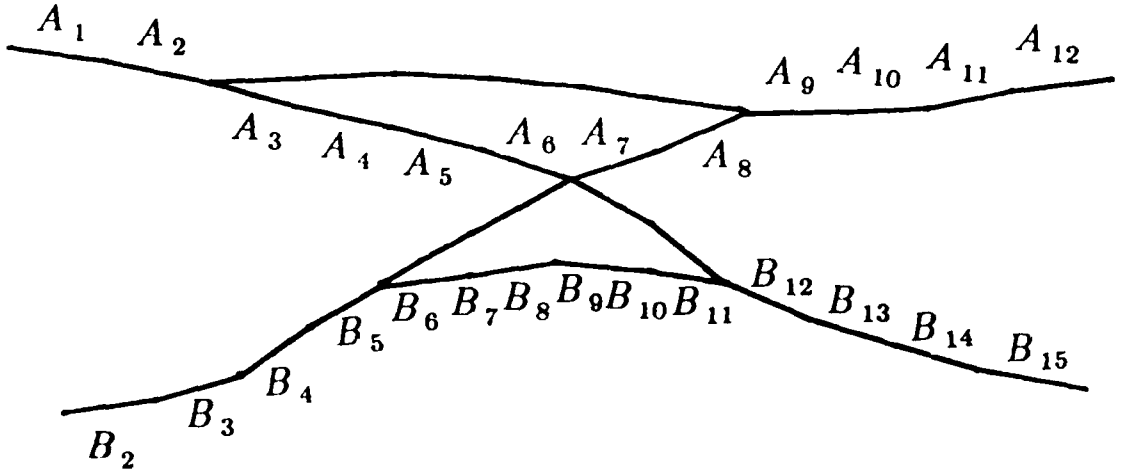
5.3.2 Communication Schemes

A hierarchical type of communication scheme was tested:

1. Every odd scan node 1 sends information to node 2, and node 3 sends information to node 4.

2. Every even scan node 4 sends information to node 2.

i.e, node 1 and node 3 only transmit information to other nodes, node 4 is an intermediate receiver/processor/transmitter and all information is thus collected by node 2 with communication delays. The communication pattern and the information graph are shown in Figure 5-19 and Figure 5-20.



A_i : Location of Target A at scan i

B_i : Location of Target B at scan i

Figure 5-5: Target Scenario

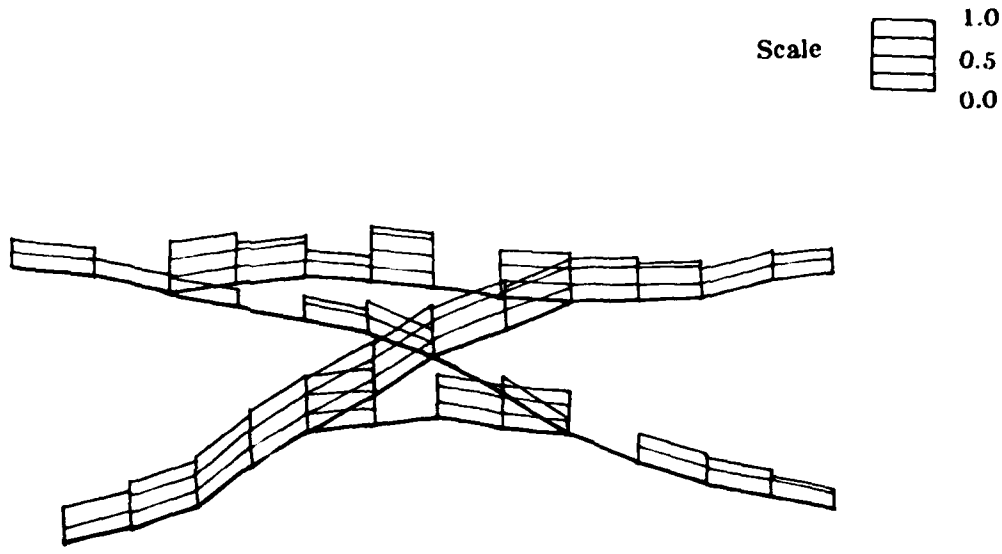


Figure 5-6: Probability of Detection of Sensor 1

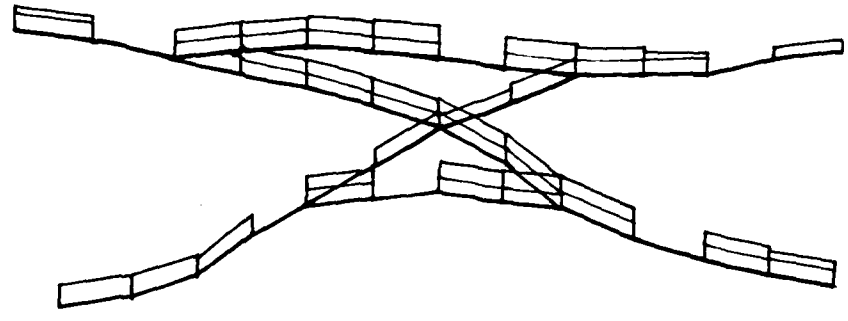


Figure 5-7: Probability of Detection of Sensor 2

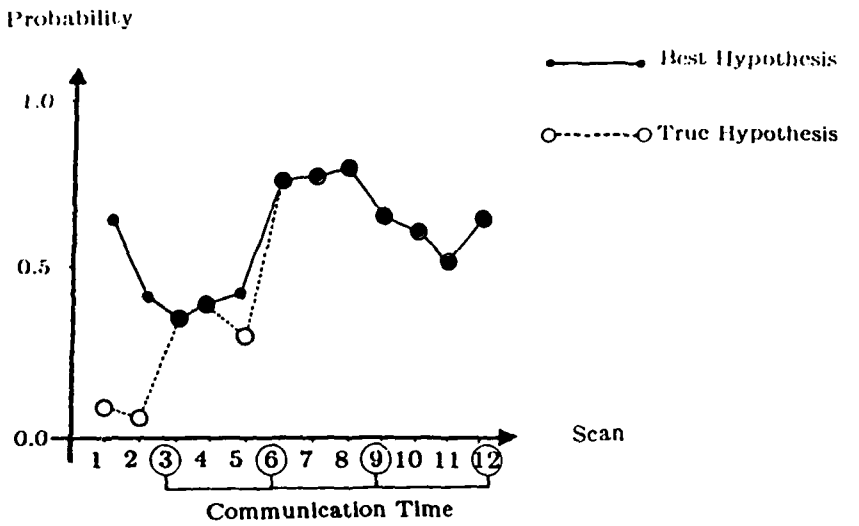


Figure 5-8: Best Hypothesis versus True Hypothesis for Node 1 in Two-Node Broadcast Case

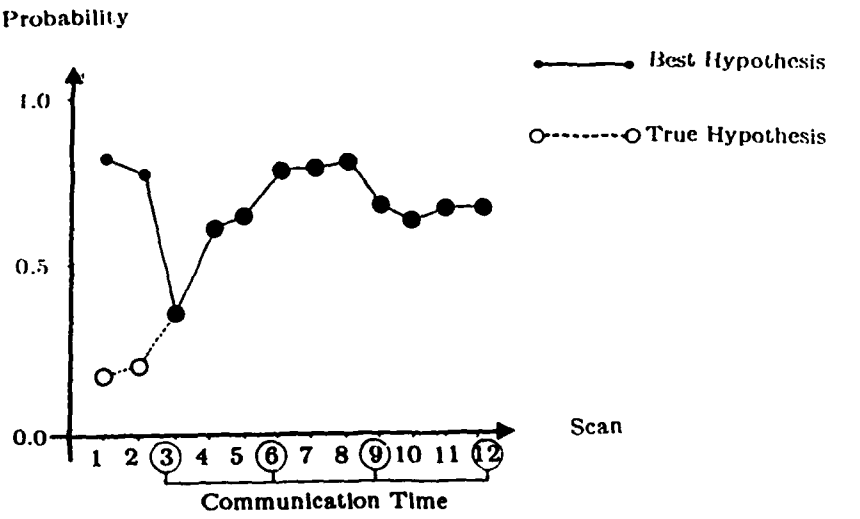


Figure 5-9: Best Hypothesis versus True Hypothesis for Node 2 in Two-Node Broadcast Case

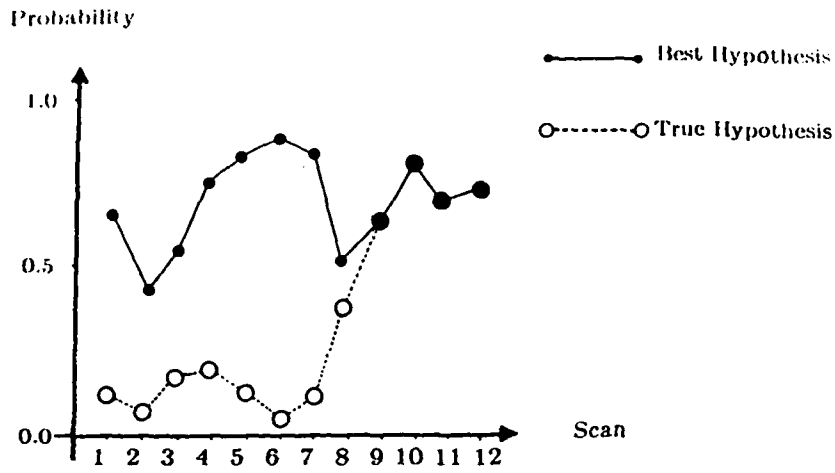


Figure 5-10: Best Hypothesis versus True Hypothesis for Node 1 in Two-Node One-Way Communication Case

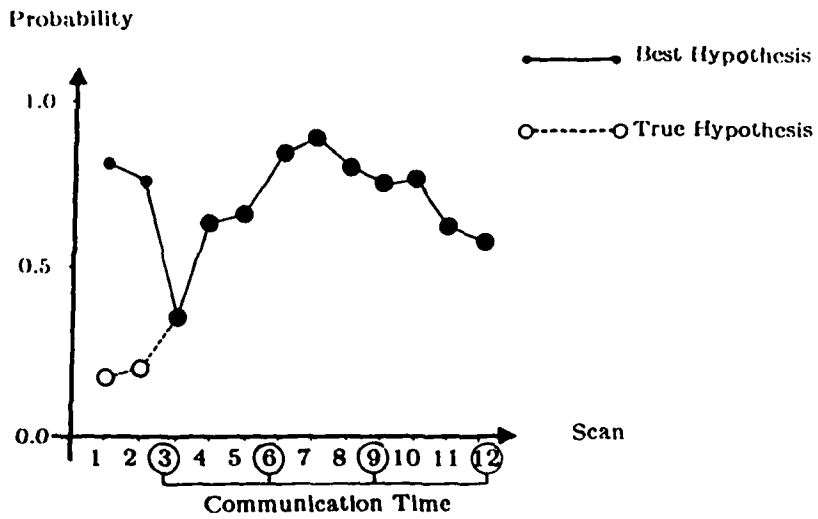


Figure 5-11: Best Hypothesis versus True Hypothesis for Node 2 in Two-Node One-Way Communication Case

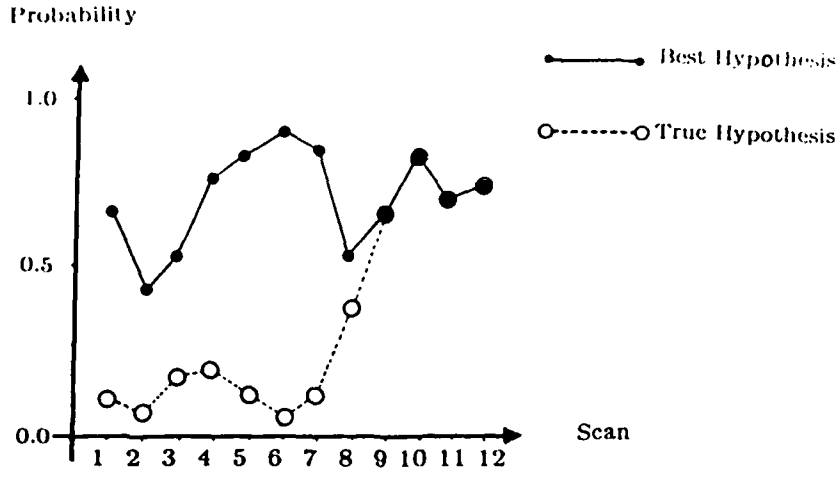


Figure 5-12: Best Hypothesis versus True Hypothesis for Node 1 in Decentralized Case

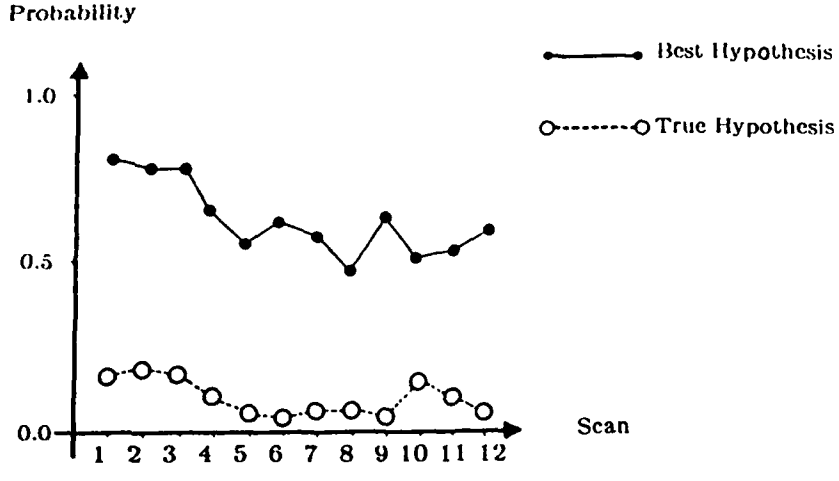


Figure 5-13: Best Hypothesis versus True Hypothesis for Node 2 in Decentralized Case

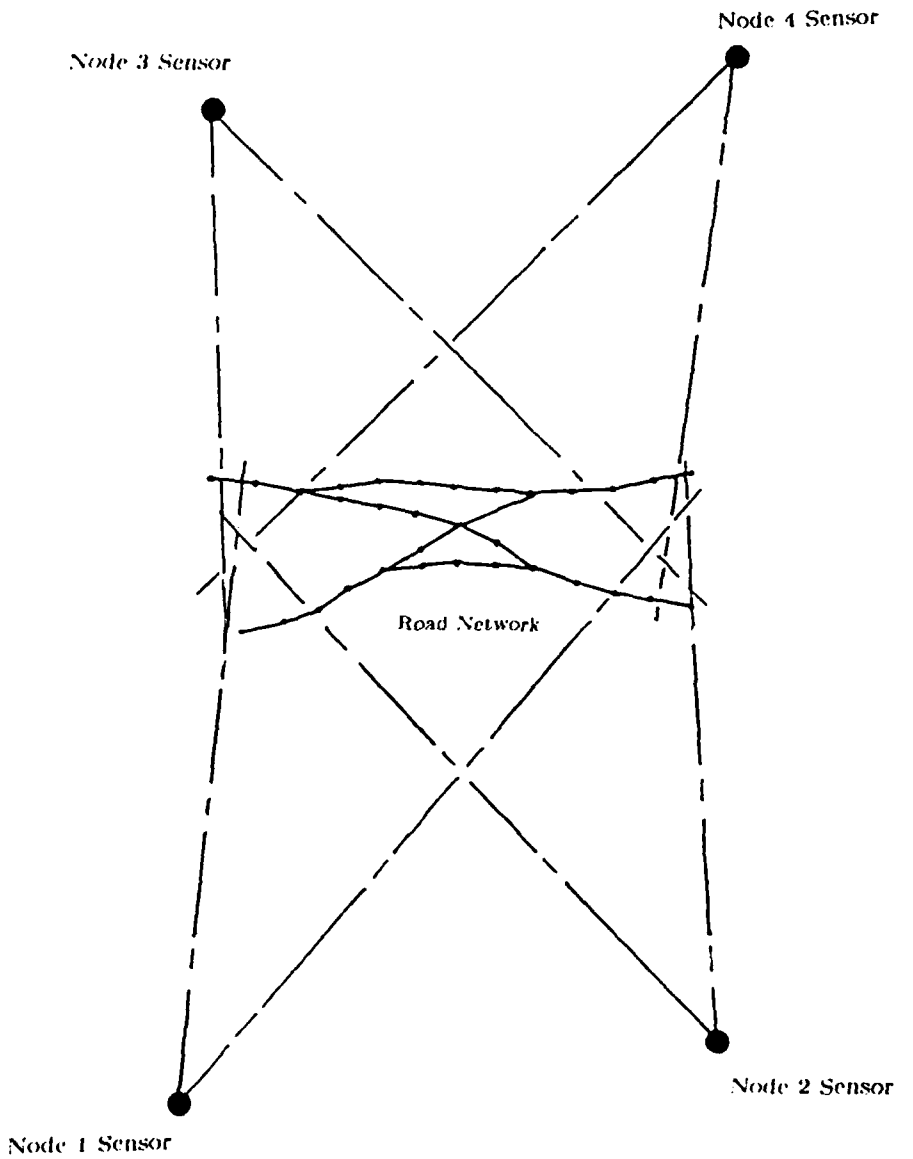


Figure 5-14: A Four-Node Scenario

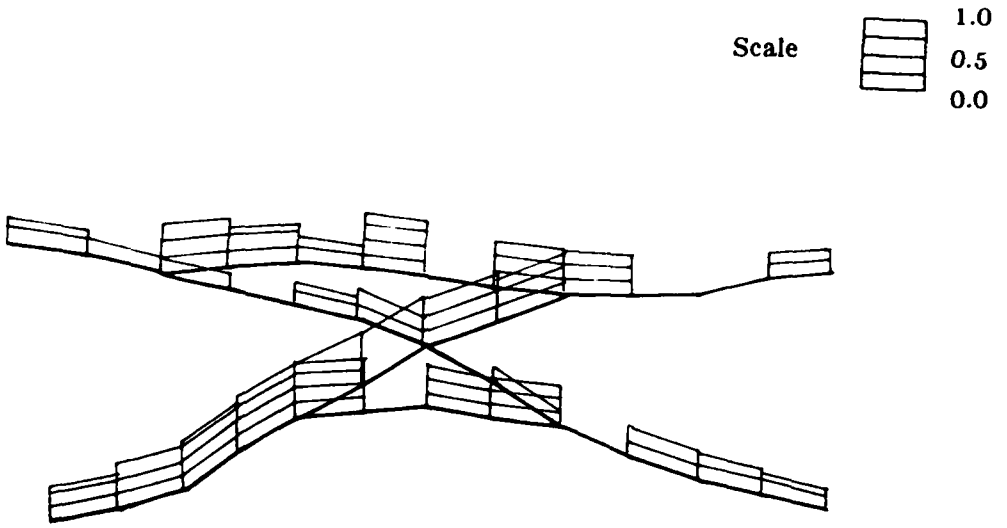


Figure 5-15: Probability of Detection of Sensor 1

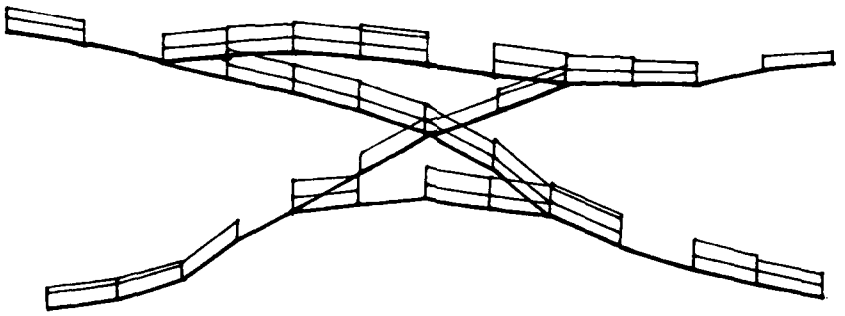


Figure 5-16: Probability of Detection of Sensor 2

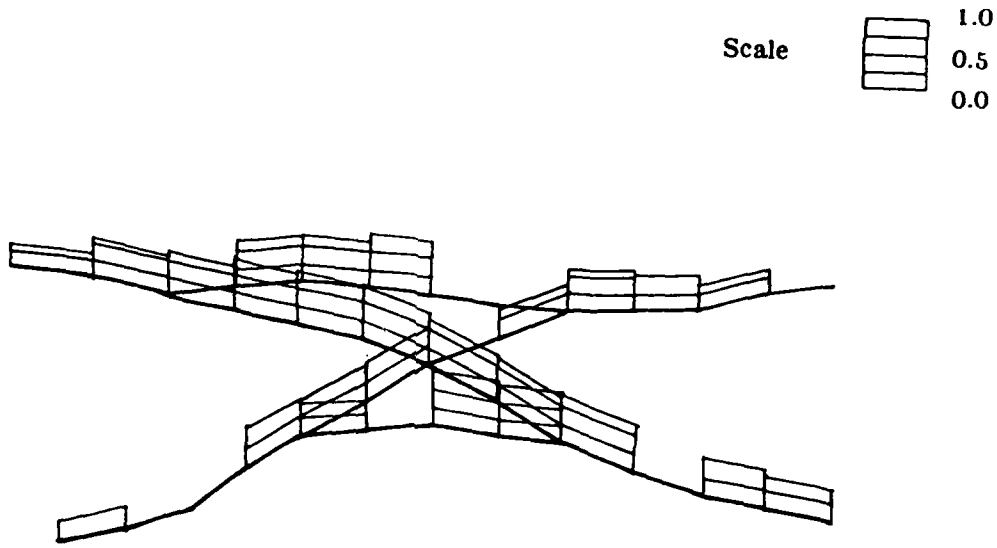


Figure 5-17: Probability of Detection of Sensor 3

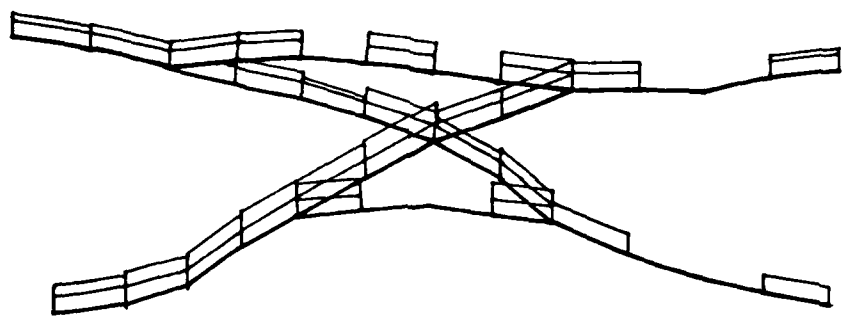


Figure 5-18: Probability of Detection of Sensor 4

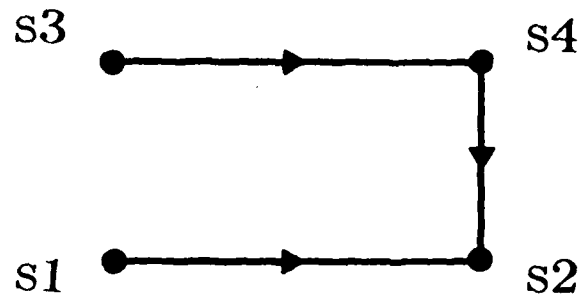


Figure 5-19: Communication Pattern of the Four-Node Case

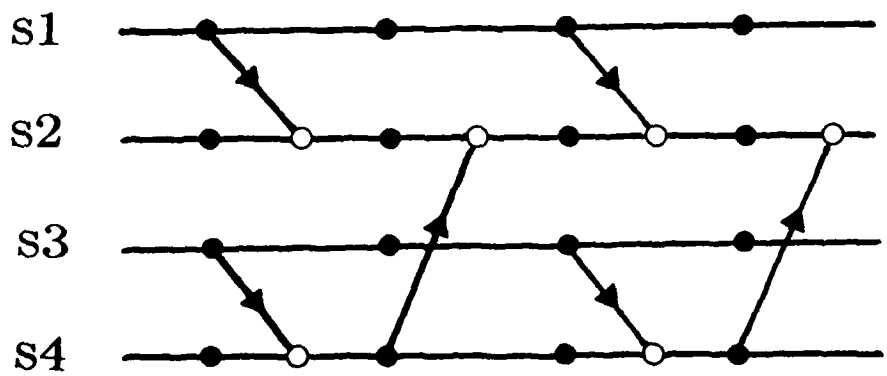


Figure 5-20: Information Graph of the Four-Node Case

5.3.3 Simulation Results

The same simulation parameters as in the previous example were used. Figure 5-21 to Figure 5-24 show the probabilities of the best hypotheses versus true hypotheses at every scan for each node. From the results we can see that node 2 and node 4 have better performance than node 1 and node 3. This is so since they have more information.

The true hypotheses of node 1 and node 3 are never higher than 0.3 (see Figure 5-21 and Figure 5-23), similar to the previous examples, this is because they only process their own local data which is not very good and never receive information from the other nodes to confirm or correct their hypotheses. Node 4 does not receive any information from node 1 or node 2, however, node 4 does receive information from node 3, which gives it fairly good sensor coverage and detections (see Figure 5-14, 5-17 and 5-18), resulting in reasonably good performance (see Figure 5-24).

From the results of node 2 (Figure 5-22), wee can see that the probability of true hypothesis approaches that of the best hypothesis after 2 scans. It is not clear from this sample run how much better the performance of node 2 compares to that of node 4. However, it is obvious that node 2 and node 4 perform better than node 1 and node 3 because of the communication structure.

5.4 CONCLUSION

The information fusion algorithms developed in Section 3 have been tested via simulations using discrete-state road network examples. Various communication schemes with different number of nodes have been examined. The simulation results have shown that the algorithms produce the expected performance.

More extensive simulations using other scenarios will be conducted after the development of the simulation environment is completed. The simulation environment itself is being enhanced to improve its flexibility and efficiency. When this development of the simulation environment is complete, we should be able to perform more extensive simulations using other scenarios. In particular,

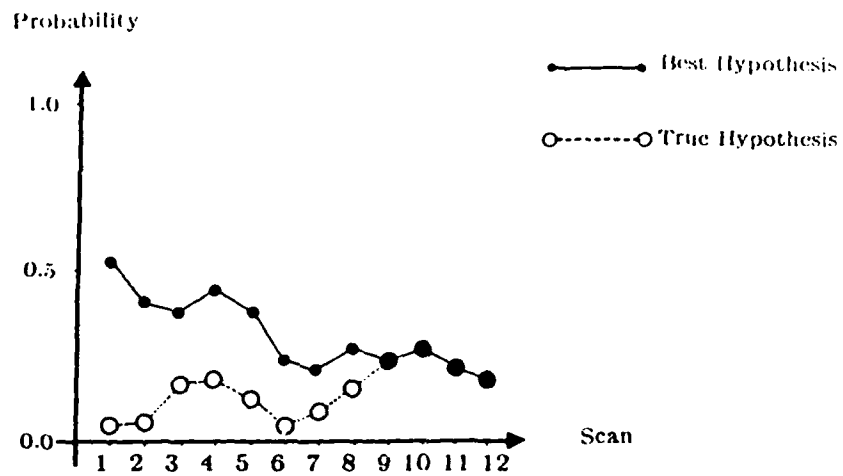


Figure 5-21: Best Hypothesis versus True Hypothesis for Node 1 in Four-Node Case

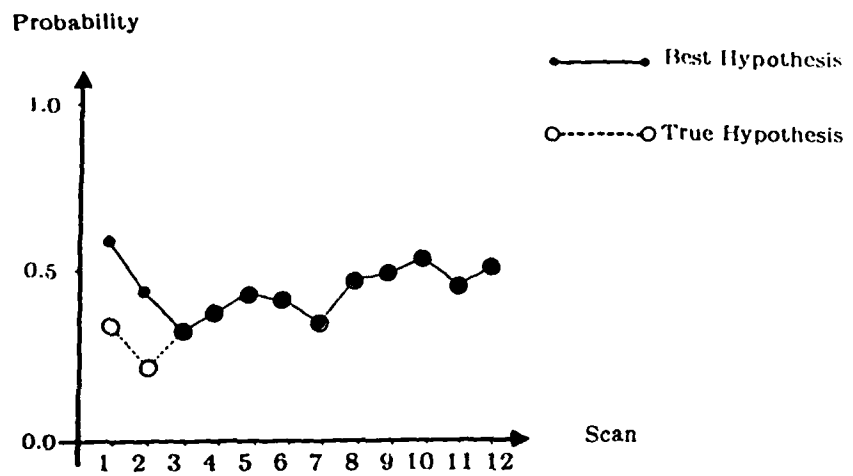


Figure 5-22: Best Hypothesis versus True Hypothesis for Node 2 in Four-Node Case

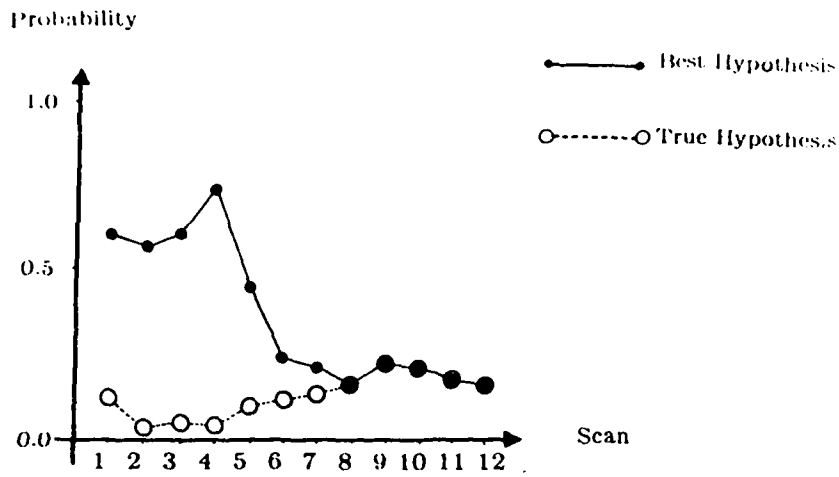


Figure 5-23: Best Hypothesis versus True Hypothesis for Node 3 in Four-Node Case

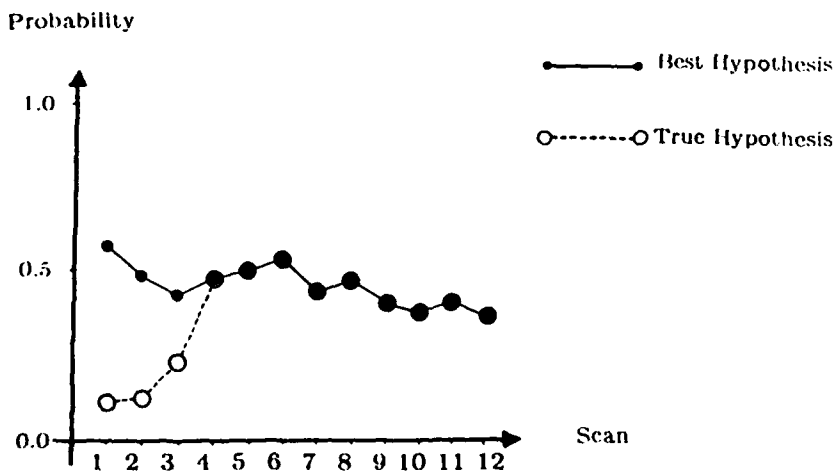


Figure 5-24: Best Hypothesis versus True Hypothesis for Node 4 in Four-Node Case

we intend to use other target and sensor models and study the effect of varying parameters such as hypothesis pruning threshold, detection probability, false alarm rate and measurement errors.

Various ways of speeding up the processing of each node as better hypothesis management, as well as ways of improving the overall system performance, such as better communication strategies, sensor control, etc. would also be investigated using this simulation environment.

REFERENECES

[1] C.Y. Chong, S. Mori, E. Tse and R.P. Wishner, "Distributed Hypothesis formation In Distributed Sensor Network," Final Report, A.I.& D.S. Technical Report TR-1015-02, Advanced Information & Decision Systems, Mountain View, Ca., May 1983.

[2] S. Mori, C.Y. Chong, E. Tse and R.P. Wishner, "Multitarget Multisensor Tracking Problems - Part I: A General Solution and a Unified View on Bayesian Approaches," A.I.& D.S. Technical Report TR-1048-01, Advanced Information & Decision Systems, Mountain View, Ca., June 1983 (revised Aug. 1984).

[3] D.B. Reid, "A Multiple Hypothesis Filter for Tracking Multiple Targets in a Cluttered Environment," Lockheed Report LMSC-D560254, Lockheed Palo Alto Research Laboratory, Palo Alto, Ca., Sept., 1977.

[4] D.B. Reid, "The Application of Multiple Target Tracking Theory to Ocean Surveillance," Proc. of 18th IEEE Conf. Decision and Contr., Fort Lauderdale, FL., pp. 1046 - 1052, Dec. 1979.

[5] H.L. Wiener, W.W. Willman, I.R. Goodman and J.H. Kullback, "Naval Ocean Surveillance Correlation Handbook, 1978," NRL Report 8340, Naval Research Lab., Washington D.C., 1979.

[6] E. Flad, "Track Correlation for Aircraft Flying in Group Formation," *Siemens Forsch. -u. Entwickl. -Ber.*, Bd. 2, Nr. 6, 1973.

[7] S. Hale *et al.*, "Algorithm for Automatic Correlation," ATLAS Working Paper #13, P.A.& R.C., April 1978.

[8] R.J. Drazovich, S. Brooks and S. Foster, "Knowledge Based Ship Classification," Proc. of Workshop of Image Understanding and Spacial Processing to Radar Signals for Automatic Ship Classification, New Orleans, La., Feb 1979

[9] D.S. Spain, G.L. Orr, C.A. O'Reilly and R.J. Drazovich, "Automated Multi-Sensor Unit Identification," Final Report, A.I.& D.S. Technical Report TR-3030-02, Advanced Information & Decision Systems, June 1984.

[10] M.R. Fehling, R. Fung, J. Rosenschein, J. Finger, B. d'Ambrosio and B.P. McCune, "Distributed Performance Maintenance for Ballistic Missile Defense," A.I.& D.S. Technical Report TR-3054-1, Advanced Information & Decision Systems, Mountain View, Ca., June 1984.

[11] J.M. Abram, C.Y. Chong, M.R. Fehling, J.S. Rosenschein and E. Tse, "Distributed Decision Making Environment," Final Report, A.I.& D.S. Technical Report TR-3013-3, Advanced Information & Decision Systems, Mountain

View, Ca., April 6, 1984.

[12] F. Hayes-Roth, D.A. Waterman and D.B. Lenat (ed.), *Building Expert Systems*, Addison-Wesley, Reading, Ma., 1983.

[13] A. Barr and E.A. Feigenbaum, *The Handbook of Artificial Intelligence - Volume I*, William-Kaufmann, Inc., L.A., Ca., 1982.

[14] R.C. Schank and R.P. Abelson, *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum, Hillsdale, N.J., 1977.

[15] A. Kay and A. Goldberg, "Personal Dynamic Media," *Computer*, Vol. 10, pp.31-41, 1977.

[16] L.D. Erman, F. Hayes-Roth, V.R, Lesser and D.R. Reddy, "The HERSAY-2 Speech-Understanding System : Integrating Knowledge to Resolve Uncertainty," *Computing Survey*, Vol. 12, No. 2, 1980.

[17] H.P. Nii and N. Aiello, "A Knowledge-Based Program for Building Knowledge-Based Programs," Proc. of Sixth International Joint Conference on Artificial Intelligence, pp. 645-655, Tokyo, 1979.

[18] R. Davis, "Applications of Meta-Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases", Report No. STA-CS-76-564, Dept. of Computer Science, Stanford Univ., 1976.

Appendix: Distributed Estimation and Multitarget Tracking

This appendix summarizes all the formal arguments concerning distributed estimation and distributed multitarget tracking in the distributed sensor network context.

1. Formal Definition of Distributed Sensor Network

We first formalize the concept of the distributed sensor network in terms of an information graph. Let N be a nonempty finite set of *information processing nodes* (simply *nodes* or *estimation agents*) and S be a nonempty finite set of *sensors* (or *information sources*). A nonempty set $\mathbf{Y}_s$, called *sensor format space* (or *information format space*), is given for each s in S. For each n in N, a set S_n of *private sensors* is given. Each S_n is a subset of S. Then, for a given interval $T=[t_0,t_1]$, a subset $\mathbf{Z}$ of the set

$$\bigcup_{s\in S}\mathbf{Y}_s\times T\times\{s\}$$

is called the *total information* (obtained in T). Each element (z,t,s) in total information $\mathbf{Z}$ is called a *data set* (or *sensor report*) *generated by sensor* s *at time* t. Any subset Z of $\mathbf{Z}$ is called *partial information set* or simply *information set*. Given the total information $\mathbf{Z}$, the set

$$\mathbf{K}=\{(t,s)\in T\times S \mid (z,t,s)\in\mathbf{Z} \text{ for } \textit{some } z\}$$

is called *total index set* for $\mathbf{Z}$. Any subset of K of $\mathbf{K}$ is called *partial index set* or simply *index set*.

A subset $\mathbf{C}$ of $T\times T\times N\times N$ is called *communication schedule*. An element (t,t',n,n') in $\mathbf{C}$ represents the transmission of a message by node n at time t which is received by node n' at time t'. We call any element of

$$\mathbf{I} = \mathbf{I}_{ST} \cup \mathbf{I}_{SR} \cup \mathbf{I}_{CT} \cup \mathbf{I}_{CR}$$

an *information node*, where

$$\mathbf{I}_{ST} = \mathbf{K} \times \{ST\}$$

$$\mathbf{I}_{SR} = \mathbf{K} \times \{SR\}$$

$$\mathbf{I}_{CT} = \{(t,n,CT) \mid (t,t',n,n') \in \mathbf{C}\}, \quad \text{and}$$

$$\mathbf{I}_{CR} = \{(t,n,CR) \mid (t',t,n',n) \in \mathbf{C}\}.$$

We call an information node in $\mathbf{I}_{ST}$ ($\mathbf{I}_{SR}$, $\mathbf{I}_{CT}$ or $\mathbf{I}_{CR}$, respectively) a *sensor transmitting node* (*sensor receiving node, communication transmitting node, communication receiving node*, reps.). An *information structure* is a binary relation $\leq$ on $\mathbf{I}$, which is reflexive and antisymmetric. $i < i'$ means $i \leq i'$ but $i \neq i'$. $i \vdash i'$ means i is an immediate predecessor of i'. The ordered set $(\mathbf{I}, \leq)$ is called an *information graph*.

Let us define

$$\mathbf{K}^{(n)} = \{(t,s) \in \mathbf{K} \mid s \in S_n\},$$

$$\mathbf{I}_{ST}^{(n)} = \mathbf{K}^{(n)} \times \{ST\}, \quad \mathbf{I}_{SR}^{(n)} = \mathbf{K}^{(n)} \times \{SR\},$$

$$\mathbf{I}_{CT}^{(n)} = \{(t,n,CT) \in \mathbf{I}_{CT} \mid t \in T\},$$

$$\mathbf{I}_{CR}^{(n)} = \{(t,n,CR) \in \mathbf{I}_{CR} \mid t \in T\}, \quad \text{and}$$

$$\mathbf{I}^{(n)} = \mathbf{I}_{SR}^{(n)} \cup \mathbf{I}_{CT}^{(n)} \cup \mathbf{I}_{CR}^{(n)}.$$

Then the following assumptions are made on the network:

Basic Assumptions

[A1] (No Sensor Sharing) $S_n \cap S_{n'} = \emptyset$ for all $(n,n') \in N \times N$ such that $n \neq n'$.

[A2] (Finiteness) $\mathbf{Z}$ and $\mathbf{C}$ are both finite.

[A3] (No Indexing Confusion) $\mathbf{Z}$ is isomorphic to $\mathbf{K}$.

[A4] (Information Sources) Each (k,ST) in $\mathbf{I}_{ST}$ is minimum in $\mathbf{I}$ and has only one

successor so that $(k,ST) \mapsto (k,SR)(\in \mathbf{I}_{SR})$

[A5] (Information Exchanges) $(t,n,CT) \mapsto (t',n',CR)$ if and only if $(t,t',n,n') \in \mathbf{C}$.

[A6] (Causality 1) $(m,t,l) \leq (m',t',l')$ implies $t \leq t'$.

[A7] (Causality 2) The binary relation $\leq$ is transitive.

[A8] (Total Order) For each n in N, $(\mathbf{I}^{[n]},\leq)$ is totally ordered.

Remarks:

(1) [A1] to [A5] are more or less standard assumptions.

(2) [A4] implies no delay in sensor reporting.

(3) [A5] allows us to model communication delays.

(4) [A6] prohibits any negative time delay.

(5) [A7] prohibits any cycle in $(\mathbf{I},\leq)$.

(6) [A8] implies sequential data processing (and implicitly perfect memory) for each node.

Given an information graph $(\mathbf{I},\leq)$, we can assign a maximum amount of information to each information node $i \in \mathbf{I}$. More precisely, for each $i \in \mathbf{I}$,

$$K^{(i)} = \{k \in \mathbf{K} \mid (k,ST) \leq i\} \tag{1}$$

is called the *maximum index set* at node i. Similarly

$$Z^{(i)} = \{(z,k) \in \mathbf{Z} \mid k \in K^{(i)}\} \tag{2}$$

is called the *maximum information set* at i. Then the following observations follow immediately from the definitions:

Proposition 1:

(1) $K^{(i)} = \{k \in \mathbf{K} \mid (k,SR) \leq i\}$ *for all* i *in* $\mathbf{I}$.

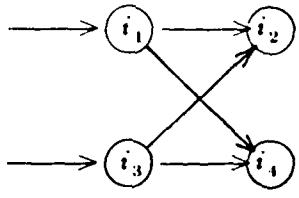
(2) $K^{(i)} \subseteq K^{(i')}$ *if* $i \leq i'$.

(3) $K^{(i)} = \bigcup_{i' < i} K^{(i')}$ *for all* i *in* $\mathbf{I}$.

(4) $K^{(i)} = \bigcup_{i' \leq i} K^{(i')}$ *for all* i *in* $\mathbf{I}$.

The proof is obvious.

Remark: The converse of the (2) of *Proposition 1* is not necessarily true as can been seen from the following counterexample:



$K^{(i_2)}=K^{(i_4)}$ but neither $i_2 \leq i_4$ or $i_4 \leq i_2$.

Due to the isomorphism between **Z** and **K**, all the K's in Proposition 1 can be replaced by Z's, i.e., we have

Proposition 2:

(1) $Z^{(i)} = \{(z,k) \in \mathbf{Z} \mid (k,SR) \leq i\}$ *for all* i *in* **I**.

(2) $Z^{(i)} \subseteq Z^{(i')}$ *if* $i \leq i'$.

(3) $Z^{(i)} = \bigcup_{i' \leq i} Z^{(i')}$ *for all* i *in* **I**.

(4) $Z^{(i)} = \bigcup_{i' \vdash i} Z^{(i')}$ *for all* i *in* **I**.

In the following, we describe two lemmas which are direct consequences of the defintions and are used in the following sections.

Lemma 1: For any set I of information nodes, we have

$$\bigcap_{i \in I} K^{(i)} = \bigcup \{K^{(i')} \mid i' \leq i \text{ for all } i \in \mathbf{I}\}$$

Proof: Suppose $k \in \bigcap_{i \in I} K^{(i)}$. Then by definition $i' = (k,SR) \leq i$ for all $i \in I$, or $k \in K^{(i')}$ and $i' \leq i$ for all $i \in I$. The converse is obvious. *Q.E.D.*

According to this lemma, given a set I of information nodes, the intersection of the maximum index sets, or equivalently common information contained in a collection of the maximum information obtained in a set of information

nodes, is characterized as the union of maximum index sets on the set of common predecessor information nodes. A stronger version of *Lemma 1* is the following:

Lemma 2: Suppose I_0 be a set of information nodes such that

(1) $\#(I_0) > 1$, [1]

(2) $\bigcap_{i \in I_0} K^{(i)} \neq \emptyset$, *and*

(3) *there is no pair* $(i, i') \in I_0 \times I_0$ *such that* $i < i'$.

Then,

$$I_1 = \{i_1 \in \mathbf{I}_{CT} \mid i_1 < i_0 \text{ for all } i_0 \in \mathbf{I}_0\}$$

is not empty and we have

$$\bigcap_{i \in I_0} K^{(i)} \quad \bigcup_{i \in I_1} K^{(i)} \tag{5}$$

Proof: $I_1 = \emptyset$ implies $\bigcup_{i \in I_1} K^{(i)} = \emptyset$. Therefore it suffices to prove (5). As in *Lemma 1*, it follows directly from the definitions that $\bigcup_{i \in I_1} K^{(i)}$ is a subset of $\bigcap_{i \in I_0} K^{(i)}$

To prove the other inclusion, suppose $k \in \bigcap_{i_0 \in I_0} K^{(i_0)}$ or $(k, SR) \leq i_0$ for all $i_0 \in I_0$. Let $k = (s, t) \in S_n \times T$ where $n \in N$ is the node uniquely (*Assumption* [A1]) determined by k. Since $\mathbf{I}^{(n)}$ is totally ordered (*Assumption* [A8]), assumptions (1) and (3) guarantee existence of an i'' in $I_0 \cap \mathbf{I}^{(n')}$ for some $n' \in N$ such that $n' \neq n$. Hence, there exists a transmission node i' which belongs to n, i.e., $i' \in \mathbf{I}_{CT}^{(n)}$, such that

$$(k, SR) < i' < i'' \tag{6}$$

for some i'' in I_0. Let $\bar{i}$ be the smallest element i' in $\mathbf{I}_{CT}^{(n)}$ which satisfies (6) with some i'' in I_0.

---

1 For any set A, $\#(A)$ is its cardinality, or the number of its elements.

Then, for any $i_0 \in I_0 \backslash \mathbf{I}^{(n)}$ we must have $\bar{i} < i_0$ because (k, SR) must be connected through a communication link which includes $\bar{i}$. If $i_0 \in I_0 \cap \mathbf{I}^{(n)}$, we must also have $\bar{i} < i_0$ because $\mathbf{I}^{(n)}$ is totally ordered and $i_0 \leq \bar{i}$ contradicts with assumption (3). *Q.E.D.*

2. Distributed Estimation Problems

This section discusses the distributed estimation problems with arbitrary information graphs.

2.1 Static Estimation Problems The object to be estimated is modeled as a random element x (called *state-of-the-world* or simply *state*) taking values in a measurable space $(\mathbf{X}, \mathbf{B})$ where $\mathbf{X}$ is a locally compact Hausdorff space satisfying the second axiom of countability[1] and $\mathbf{B}$ is the *σ-field* of Borel sets in $\mathbf{X}$. Let μ be a *σ-finite* measure on $(\mathbf{X}, \mathbf{B})$ and call the measure space $(\mathbf{X}, \mathbf{B}, \mu)$ *state space*. Implicitly we are assuming an underlying probability space $(\Omega, \mathbf{F}, Prob)$ but we rarely must refer to it.

In parallel, for each sensor $s \in S$, $\mathbf{Y}_s$ is a locally compact Hausdorff space satisfying the second axiom of countability and the sensor format space is redefined as the measure space $(\mathbf{Y}_s, \mathbf{B}_s, \mu_s)$ where $\mathbf{B}_s$ is the *σ-field* of Borel sets in $\mathbf{Y}_s$ and μ_s is a *σ-finite* measure on $(\mathbf{Y}_s, \mathbf{B}_s)$. All the sets defined in the previous section are then redefined as random sets. For each sample, we assume that all the assumptions [A1] - [A8] are satisfied (at least with probability one). According to the common custom, we will abuse the notations $p(\,)$ and $p(\cdot \mid \cdot)$ to the full extent. We will list our assumptions below:

Assumptions:

[B1] (Absolute Continuity) Random variable x has an absolutely continuous dis-

---

1 This is only necessary because we can use conditional probabilities and their densities etc. freely without risk.

tribution.

[B2] (Independence) Random sets **K** and **C** and random ordered set $(\mathbf{I}, \leq)$ are independent of state x.

[B3] (Conditional Independence and Absolute Continuity) Given x and **K**, each element in **Z** is conditionally independent from each other and has an absolutely continuous conditional distribution.

To obtain a general result with an arbitrary information graph, we need a couple of lemmas.

Lemma 9: Let m be a positive integer and let $(a_i)_{i=1}^m \in (-\infty,\infty)^m$. For any subset M of $\mathbf{M}=\{1,\ldots,m\}$, define $a[\emptyset]=0$ and, if $M\neq\emptyset$,

$$a[M] = \sum_{m\in M} a_m .$$

*Then, for any positive integer n and for any n subsets, $M_1, \cdots, M_n$, of **M**, we have*

$$a[\bigcup_{i=1}^{n} M_i] = \sum_{i=1}^{n} (-1)^{i-1} \Big(\sum_{N\in \mathbf{N}_i^n} a[\bigcap_{j\in N} M_j] \Big) \tag{9}$$

where

$$\mathbf{N}_i^n = \{N \subseteq \{1,\ldots,n\} \mid \#(N)=i\} \tag{10}$$

Proof: We will use the mathematical induction. When $n=1$, the statement is clearly true. When $n=2$, (9) is

$$a[M_1 \cup M_2] = a[M_1] + a[M_2] - a[M_1 \cap M_2] \tag{11}$$

which is also obviously true for any two subsets M_1 and M_2 of **M**. Suppose the statement of the lemma is true for $n-1$ when $n>2$. Then, using (11), we have

$$a[\bigcup_{i=1}^{n} M_i] = a[\bigcup_{i=1}^{n-1} M_i] + a[M_n] - a[\bigcup_{i=1}^{n-1}(M_i \cap M_n)]$$

$$= \sum_{j=1}^{n} a[M_j]$$

$$+ \sum_{i=2}^{n-1} (-1)^{i-1} \Big(\sum_{N \in \mathbf{N}_i^{n-1}} a[\bigcap_{j \in N} M_j] + \sum_{N' \in \mathbf{N}_{i-1}^{n-1}} a[(\bigcap_{j \in N'} M_j) \cap M_n] \Big)$$

$$+ (-1)^{n-1} a[\bigcap_{i=1}^{n} M_j]$$

$$= \sum_{i=1}^{n} (-1)^{i-1} \Big(\sum_{N \in \mathbf{N}_i^n} a[\bigcap_{j \in N} M_j] \Big)$$

which proves (9) for n. *Q.E.D.*

For any information index k in $\mathbf{K}$, by z_k we mean the random element in $\mathbf{Y}_s$ such that $(z_k, k) = (z_k, s, t) \in \mathbf{Z}$ is uniquely determined by k.

Lemma 4: For any partial information set Z whose index set is K, we have

$$p(x \mid Z)\, p(Z \mid K) = \Big(\prod_{k \in K} p(z_k \mid x)\Big)\, p(x) \tag{12}$$

The proof of this lemma is a direct consequence of Assumptions [B1] - [B3]. In order to avoid complication, we will assume that any term in (12) is strictly positive (at least in an appropriate a.e. notion).

The following two theorems are the main results of distributed estimation for fusing the estimates at various nodes to obtain a global estimate.

Theorem 1: For every positive integer n and for any n partial informations sets, $Z_1, \ldots, Z_n$, we have

$$p(x \mid \bigcup_{i=1}^{n} Z_i) = c \prod_{i=1}^{n} \Big(\prod_{N \in \mathbf{N}_i^n} p(x \mid \bigcap_{j \in N} Z_j) \Big)^{(-1)^{i-1}} \tag{13}$$

where

$$c = p(\bigcup_{i=1}^{n} Z_i \mid \bigcup_{i=1}^{n} K_i)^{-1} \prod_{i=1}^{n} \Big(\prod_{N \in \mathbf{N}_i^n} p(\bigcap_{j \in N} Z_j \mid \bigcap_{j \in N} K_j) \Big)^{(-1)^{i-1}} \tag{14}$$

Proof: For any partial information set Z with index set K, we define $a[Z] = \prod_{k \in K} p(z_k \mid x)$. Then Lemma 4 implies

$$a[Z] = p(Z)\, p(x \mid Z)\, p(x)^{-1} \tag{15}$$

Using the product version of Lemma 3 and applying (15) for each Z_i, we have

$$\begin{aligned} a[\bigcup_{i=1}^{n} Z_i] &= \prod_{i=1}^{n} \Big(\prod_{N \in \mathbf{N}_i^n} a[\bigcap_{j \in N} Z_j] \Big)^{(-1)^{i-1}} \\ &= \prod_{i=1}^{n} \Big(\prod_{N \in \mathbf{N}_i^n} p(\bigcap_{j \in N} Z_j \mid \bigcap_{j \in N} K_j) \Big)^{(-1)^{i-1}} \\ &\quad \cdot \prod_{i=1}^{n} \Big(\prod_{N \in \mathbf{N}_i^n} p(x \mid \bigcap_{j \in N} Z_j) \Big)^{(-1)^{i-1}} \\ &\quad \cdot \prod_{i=1}^{n} \Big(\prod_{N \in \mathbf{N}_i^n} p(x)^{-1} \Big)^{(-1)^{i-1}} \end{aligned}$$

The last term becomes $p(x)^{-1}$ since $\sum_{i=1}^{n} (-1)^{i-1} \binom{n}{i} = 1$. Applying (15) to $\bigcup_{i=1}^{n} Z_i$, we have (13) and (14). *Q.E.D.*

Unfortunately, this theorem is not sufficient to describe a general information fusion algorithm since (13) may contain probability densities conditioned by intersections of Z_i's. However, these intersections can be decomposed further by *Lemma 2* in the previous section. The result can be stated in the following theorem. For notational convenience, we add an extra element $i_\emptyset$ to the set I of all the information nodes and let $\bar{\mathbf{I}} = \mathbf{I} \cup \{i_\emptyset\}$. Then $(\bar{\mathbf{I}}, \leq)$ is constructed by letting $i_\emptyset$ be the immediate predecessor of all the minimum nodes in the original information graph $(\mathbf{I}, \leq)$. Then we have $Z^{(i_\emptyset)} = K^{(i_\emptyset)} = \emptyset$.

Theorem 2: For any communication receiving node i_0 in $\mathbf{I}_{CR}$, there exists a subset $\bar{I}$ of set $\bar{\mathbf{I}}$ of extended information nodes such that $\bar{I} < i_0$ (This means $\bar{i} < i_0$ for all $\bar{i}$ in $\bar{I}$.) and

$$p(x \mid \bigcup_{i \in I} Z^{(i)}) = c \prod_{\bar{i} \in \bar{I}} p(x \mid Z^{(\bar{i})})^{\alpha(\bar{i})} \tag{16}$$

where I is the set of immediate predecessors of i_0, $(\alpha(\bar{i}))_{\bar{i} \in \bar{I}}$ is some indexed tuple such that $\alpha(\bar{i})$ is a nonzero integer for each $\bar{i}$, and c is the normalizing constant defined by

$$c = p(\bigcup_{i \in I} Z^{(i)} \mid \bigcup_{i \in I} K_{(i)})^{-1} \prod_{\bar{i} \in \bar{I}} p(Z^{(\bar{i})} \mid K^{(\bar{i})})^{\alpha(\bar{i})} \tag{17}$$

Proof: It follows directly from *Theorem 1* that

$$p(x \mid \bigcup_{i \in I} Z^{(i)}) = c' \prod_{m=1}^{\#(I)} \Big(\prod_{J \in \mathbf{I}_m} p(x \mid \bigcap_{j \in J} Z^{(j)}) \Big)^{(-1)^{m-1}} \tag{18}$$

where c' is an appropriate normalizing constant and $\mathbf{I}_m = \{J \subseteq I \mid \#(J) = m\}$. When some J in (18) is not a singleton, we have $\bigcap_{j \in J} Z^{(j)} = \emptyset$ or $Z^{(i')}$ for some i' in $\mathbf{I}$, or otherwise, by *Lemma 2*, there exists a $J' \subseteq \mathbf{I}$ such that $J' < J$ and $\bigcup_{j' \in J'} Z^{(j')} = \bigcap_{j \in J} Z^{(j)}$. Then we can substitute this factor using *Theorem 1*. Repeating this process as necessary but finitely many times, each factor in the right hand side of (18) can be reduced to factors each of which is conditioned by $Z^{(i')}$ for some $i' \leq I$ or $\emptyset$ (the *a priori* distribution). When we cancel the factors and gather identical factors, we have (16) and (17). *Q.E.D.*

2.2 Dynamic Estimation Problems In dynamic estimation problems, the object to be estimated is a stochastic process. Thus we redefine x as a stochastic process $(x_t)_{t \in T}$ on state space $(\mathbf{X}, \mathbf{B}, \mu)$. Since it is in general impossible to estimate x itself, as being customly in filtering problems, we are interested in estimating x_t for a fixed time t in T or $(x_{t_1}, \ldots, x_{t_n})$ for a given tuple $(t_i)_{i=1}^n$ of times in T. We must alter the assumptions [B1] - [B3] according to this new situation. Although we may handle non-markovian processes, it seems safe to

maintain markovian properties to avoid any unnecessary irregularity.

Assumptions:

[C1] (Absolute Continuity and Markovian Properties) The stochastic process $(x_t)_{t\in T}$ is a temporally homogenious Markov process with a state transition probability,

$$F_{\Delta t}(E \mid x) = Prob(\{x_{t+\Delta t}\in E\} \mid x_t = x) \tag{19}$$

for any $t\in T$, for any $\Delta t \geq 0$ such that $t+\Delta t\in T$ and for any $E\in \mathbf{B}$. For any given finite set, $\{t_1, \ldots, t_n\}$, the joint probability distribution of $(x_{t_1},\ldots,x_{t_n})$ is absolutely continuous with respect to product measure μ^n.

[C2] (Independence) Random set $\mathbf{K}$ and random ordered set $(\mathbf{I},\leq)$ are independent of $(x_t)_{t\in T}$.

[C3] (Conditional Independence and Absolute Continuity) For each (t,s) in $T\times S$, we have a transition probability density function $p(\cdot \mid \cdot)$ from the state space to the information format space such that, for any partial information Z with index set K, we have

$$p(Z \mid (x_{t_k})_{k\in K}) = p(K) \prod_{k\in K} p(z_k \mid x_{t_k}) \tag{20}$$

where $(t_k,s)=k$ and (z_k,k) is the unique element in Z for each k in K.

If $\mathbf{K}$ is known beforehand and if x in Section 2.1 is replaced by $(x_{t_k})_{k\in\mathbf{K}}$[1], *Theorems 2* and *3* are still valid when x is replaced by (x_{t_k}). This, however, requires the computation of a high dimensional probability distribution and is not very realistic. Unfortunately, unless we have make a rather strict restrictions, i.e., bi-directional determinisity, it may be impossible to have a formula which is both general and applicable.

---

1 Actually, when fusing information $Z = \bigcup_i Z_i$, $(x_{t_k})_{k\in K}$ is enough where K is the index set for Z.

Deterministic Processes: We considered a special class of deterministic processes with the following assumption:

Assumption [D] Stochastic process $(x_t)_{t\in T}$ is bi-directionally deterministic, i.e., there exists a one-parameter group of homeomorphic operators on $\mathbf{X}$, $(\Phi_t)_{t=-\infty}^{\infty}$, such that

$$Prob(\{x_t \in E\} \mid x_{t'}=x) = \chi(\Phi_{t-t'}(x);E) \tag{21}$$

for all $(t,t')\in T\times T$ ($\chi(.;A)$ is the indicator function of set A.).

With this assumption, the stochastic process is deterministic in both directions of time. Therefore, when a set of data is accumulated in the past, we can consider the present time as the initial time. We also assume that extrapolation of state probability density can be done in both directions freely. Then, we can restate Theorems 1 and 2 in Section 2.1 as follows:

Theorem 3: Let $i=(t,n,CR)$ be a communication receiving node in $\mathbf{I}_{CR}$ and $\{i_1,.....,i_M\}$ be the set of all the immediate predecessors of i. Then we have

$$p(x_t \mid \bigcup_{m=1}^{M} Z_m) = c \prod_{m=1}^{M} \Big(\prod_{(J\subseteq\{1,...,M\},\#(J)=m)} p(x_t \mid \bigcap_{j\in J} Z_j) \Big)^{(-1)^{m-1}} \tag{22}$$

where $Z_m = Z^{(i_m)}$ and

$$c = p(\bigcup_{m=1}^{M} Z_m \mid \bigcup_{m=1}^{M} K_m)^{-1} \prod_{m=1}^{M} \Big(\prod_{(J\subseteq\{1,...,M\},\#(J)=m)} p(\bigcap_{j\in J} Z_j \mid \bigcap_{j\in J} K_j) \Big)^{(-1)^{m-1}} \tag{23}$$

with $K_m = K^{(i_m)}$. Moreover, there exists a subset $\bar{I}$ of set $\bar{\mathbf{I}}$ of extended information nodes such that $\bar{I} < i$ and

$$p(x_t \mid \bigcup_{m=1}^{M} Z_m) = d \prod_{\bar{i}\in\bar{I}} p(x_t \mid Z^{(\bar{i})})^{\alpha(\bar{i})} \tag{24}$$

where $(\alpha(\bar{i}))_{\bar{i}\in\bar{I}}$ is some indexed tuple such that $\alpha(\bar{i})$ is a nonzero integer for each $\bar{i}$ and

$$d = p\left(\bigcup_{m=1}^{M} Z_m \mid \bigcup_{m=1}^{M} K_m\right)^{\dagger} \prod_{\tilde{\tau}\in T} p\left(Z^{(\tilde{\tau})} \mid K^{(\tilde{\tau})}\right)^{\alpha(\tilde{\tau})} \tag{25}$$

Proof: In the proof of *Theorem 1*, redefine $a[Z]$ as

$$a[Z] = \prod_{k\in K} p(z_k \mid \Phi_{t_k - t}(x_t)).$$

Then we can carry out the same proof as in *Theorems 2* and *3*. *Q.E.D.*

3. Distributed Multi-Target Tracking

In Appendix C of [1], distributed multi-target tracking problems were formulated assuming broadcasting-type communication. In this section, we generalize the results stated there to distributed multi-target tracking with an arbitrary information graph.

3.1 Formalism of Multi-target Tracking Problems In multi-target tracking problems, we assume stochastic processes whose states include number-of-target component and sensor data include number-of-measurement information. For this purpose, we must reformulate the state space and the sensor format spaces. In general, by a hybrid space, we mean a direct product of a measurable set (called continuous part) in a Euclidean space and a finite set (called discrete part). Let X be a hybrid space and μ be the direct product measure composed of Lebesgue measure on the continuous part and the counting measure on the discrete measure. Then we consider a stochastic process on

$$\mathbf{X} = \bigcup_{n=0}^{\infty} X^n \times \{n\} \tag{30}$$

where X^0 is a formal singleton. Thus, we write the stochastic process as

$$((x_i(t))_{i=1}^{N_T(t)}, N_T(t))_{t\in T}$$

which is actually a stochastic process on $\mathbf{X}$ which is locally compact and metrizable with a *σ-finite* measure constructed by $(\mu^n)_{n=1}^{\infty}$.

For each sensor s in S, let Y_s be a hybrid space and

$$\mathbf{Y}_s = \bigcup_{m=0}^{\infty} (Y_s)^m \times \{m\} \tag{31}$$

where $(Y_s)^0$ is again a formal singleton. Let μ_s be the hybrid measure on Y_s for each s. Then every $\mathbf{Y}_s$ is locally compact and metrizable with a *σ-finite* measure composed of $(\mu_s^m)_{m=1}^{\infty}$. Thus every element in every $\mathbf{Y}_s$ has a form $((y_i)_{i=1}^m, m)$ or $(\theta,0)$. The second element represents the number of measurements while the first element is the vector of measurement values in Y_s. For each k in $\mathbf{K}$, we may write an element in $\mathbf{Z}$ uniquely determined by k as $((y_j(k))_{j=1}^{N_M(k)}, N_M(k), k)$. We also assume, for each k in $\mathbf{K}$, a random function A_k whose domain is included in $\{1,.....N_T\}$ and whose range is included in $\{1,.....,N_M(k)\}$. $j=A_k(i)$ means that the i-th target is detected and creates the j-th measurement at k. This assumption excludes split measurements. We also assume that there is no merged measurement. This means every A_k is 1-to-1. We consider the following random sets

$$\mathbf{J} = \bigcup_{k\in\mathbf{K}} \{1,.....,N_M(k)\}\times\{k\} \tag{32}$$

and

$$\Lambda = \{\{(j,k)\in\mathbf{J} \mid j=A_k(i)\} \mid i\in\{1,....,N_T\}\} \setminus \{\emptyset\} \tag{33}$$

We call any subset J of $\mathbf{J}$ a *measurement index set*. Then the no-split/no-merged measurement assumption implies that, for any measurement index set J, the restriction of Λ on J defined by

$$\Lambda|J = \{\tau\cap J \mid \tau\in\lambda\} \setminus \{\emptyset\} \tag{34}$$

satisfies the following conditions:

(1) Each member τ of $\Lambda|J$ is a subset of J and contains at most one point in $\{1,.....N_M(k)\}\times\{k\}$ for each $k\in\mathbf{K}$.

(2) Any two sets belonging to $\Lambda|J$ do not intersect.

For each measurment index set J, define

$$\mathbf{T}(J) = \{\tau \subseteq J \mid \#(\tau \mid k) \leq 1\} \tag{35}$$

and

$$\mathbf{H}(J) = \left\{ \lambda \subseteq \mathbf{T}(J) \setminus \{\emptyset\} \;\middle|\; \begin{array}{c} \tau_1 \cap \tau_2 = \emptyset \text{ for } \textit{any} \\ (\tau_1, \tau_2) \in \lambda \times \lambda \textit{ such that } \tau_1 \neq \tau_2 \end{array} \right\} \tag{36}$$

In (35) and other subsequent equations, we use the notation

$$\tau \mid k = \tau \cap (\{1, \ldots, N_M(k)\} \times \{k\})$$

for any set τ. We call any element in $\mathbf{T}(J)$ a *track* on J and any element in $\mathbf{H}(J)$ a *hypothesis* on J. According to these definitions, for any measurement index set J, $\Lambda|J$ is in $\mathbf{H}(J)$ and each element in $\Lambda|J$ is in $\mathbf{T}(J)$.

3.2 Information Fusion Problems The distributed estimation problem in this multi-target tracking environment is to obtain (in a distributed way) $\textit{Prob}\left\{ \Lambda|J^{(i)} \mid Z^{(i)} \right\}$ for each information node i in $\mathbf{I}$ where $J^{(i)}$ is defined as

$$J^{(i)} = \{(j,k) \in \mathbf{J} \mid k \in K^{(i)}\} \tag{37}$$

This problem actually consists of two problems:

(1) (Hypothesis Formation) Suppose i_0 is a communication receiving node and I is the set of its immediate predecessor information nodes. If every node i in I contains all the set of tracks and hypotheses on $J^{(i)}$, how can we construct $\mathbf{T}(\bigcup_{i \in I} J^{(i)})$ and $\mathbf{H}(\bigcup_{i \in I} J^{(i)})$?

(2) (Hypothesis Evaluation) Suppose the above sets of tracks and hypotheses are generated. How can we evaluate each hypothesis using the evaluation made by the predecessor nodes ?

We will discuss these problems separately in the following:

Hypothesis Formation: For each track τ in $\mathbf{T}(\mathbf{J})$, each hypothesis λ in $\mathbf{H}(\mathbf{J})$ and each $J \subseteq \mathbf{J}$, we define the *restriction* of τ on J and the *restriction* of λ on J by $\tau \cap J$ and

$$\lambda|J = \{\tau \cap J \mid \tau \in \lambda\} \setminus \{\emptyset\} \quad . \tag{38}$$

respectively. Each of these restriction operations defines a partial order on $\mathbf{T}(\mathbf{J})$ or $\mathbf{H}(\mathbf{J})$ which is *arborescent* (tree-like -- for each element, the set of its predecessors is totally ordered). Moreover, the restriction operations are commutative with the intersection operation in the sense that $(\tau \cap J_1) \cap J_2 = \tau \cap (J_1 \cap J_2)$ and

$$(\lambda|J_1)\,|\,J_2 = \lambda\,|\,(J_1 \cap J_2) \tag{39}$$

for any measurement index sets, J_1 and J_2. (39) is not quite obvious but can be easily proved.

Let $(J_i)_{i \in I}$ be an arbitrary tuple of measurement index sets where I is an arbitrary nonempty set. Then we call any tuple $(\tau_i)_{i \in I}$ of tracks in $\coprod_{i \in I} \mathbf{T}(J_i)$ *fusable* if there exists a track τ in $\mathbf{T}(\bigcup_{i \in I} J_i)$ such that

$$\tau \cap J_i = \tau_i \tag{40}$$

for all $i \in I$. Similarly we call any tuple $(\lambda_i)_{i \in I}$ of hypotheses in $\coprod_{i \in I} \mathbf{H}(J_i)$ *fusable* if there exists a hypothesis λ in $\mathbf{H}(\bigcup_{i \in I} J_i)$ such that

$$\lambda|J_i = \lambda_i \tag{41}$$

for all $i \in I$. The following lemma gives a sufficient and necessary condition for each of the above fusability definitions:

Lemma 5: Let $(J_i)_{i \in I}$ be an arbitrary tuple of measurement index sets. Then we have

(1) *Any track tuple $(\tau_i)_{i \in I}$ in $\coprod_{i \in I} \mathbf{T}(J_i)$ is fusable if and only if*

$$\tau_{i_1} \bigcap (J_{i_1} \bigcap J_{i_2}) = \tau_{i_2} \bigcap (J_{i_1} \bigcap J_{i_2}) \tag{42}$$

for all $(i_1, i_2) \in I \times I$.

(2) *Any hypothesis tuple* $(\lambda_i)_{i \in I}$ *in* $\coprod_{i \in I} \mathbf{H}(J_i)$ *is fusable if and only if*

$$\lambda_{i_1} | (J_{i_1} \bigcap J_{i_2}) = \lambda_{i_2} | (J_{i_1} \bigcap J_{i_2}) \tag{43}$$

for all $(i_1, i_2) \in I \times I$.

Proof: (1) and (2) state that a tuple of tracks (or hypotheses) is fusable if and only if they share the common predecessors in the overlapping measurement index set, i.e.,

$$\bar{J} = \bigcup \{ J_{i_1} \bigcap J_{i_2} \mid (i_1, i_2) \in I \times I \text{ such that } i_1 \neq i_2 \} \tag{44}$$

In (1) and (2), the "only if" parts are obvious. To prove the "if" part of (1), let $\tau = \bigcup_{i \in I} \tau_i$ and $J = \bigcup_{i \in I} J_i$. Since every τ_i is a subset of J_i, $\tau \subseteq J$. Let K be the index set uniquely determined by J. Suppose there exists a k in K such that $\#(\tau \mid k) > 1$. Then, since $1 < \#(\tau \mid k) \leq \sum_{i \in I} \#(\tau_i \mid k)$ and since each τ_i is a track on J_i, there exists $(i_1, i_2) \in I \times I$ such that $\#(\tau_{i_1} \mid k) = \#(\tau_{i_2} \mid k) = 1$ and $\tau_{i_1} \neq \tau_{i_2}$. In other words, τ_{i_1} and τ_{i_2} do not share a common predecessor on $\{1, \ldots, N_M\} \times \{k\} \subseteq J_{i_1} \bigcap J_{i_2}$, which contradicts with the assumption. Therefore, we must have $\#(\tau \mid k) \leq 1$ for all k in K, implying $\tau \in \mathbf{T}(J)$.

Since every τ_i is included in τ and in J_i, we have $\tau_i \subseteq \tau \bigcap J_i$ for every i in I. Suppose there exists an i in I such that $\tau_i \subset \tau \bigcap J_i$, i.e., there exists $(j, k) \in \tau \bigcap J_i$ such that $(j, k) \notin \tau_i$. Then, since (j, k) is in τ, there must be i' in I such that $i \neq i'$ and $(j, k) \in \tau_{i'}$, which implies that τ_i and $\tau_{i'}$ do not share a common predecessor on $\{1, \ldots, N_M(k)\} \times \{k\}$ and contradicts the assumption. We must have $\tau \bigcap J_i = \tau_i$ for all i in I, i.e., $(\tau_i)_{i \in I}$ is fusable.

To prove the "if" part of (2), we assume $\#(I)=2$. For $\#(I)>2$, we can prove it similarly but need much more complicated notations and space. Suppose $(\lambda_{i_1},\lambda_{i_2})\in\mathbf{H}(J_{i_1})\times\mathbf{H}(J_{i_2})$ and (43) holds with $I=\{i_1,i_2\}$. Let $\bar{\lambda}$ be the common predecessor on $\bar{J}=J_{i_1}\cap J_{i_2}$. Then, as shown in *Lemma 1* of Appendix C of [1], for each $\bar{\tau}\in\bar{\lambda}$ and for each i in I, there exits a unique track $\tau_i(\bar{\tau})$ which is an extension of $\bar{\tau}$, i.e., $\tau_i(\bar{\tau})\cap\bar{J}=\bar{\tau}$. Let

$$\lambda = \{\bigcup_{i\in I}\tau_i(\bar{\tau}) \mid \bar{\tau}\in\bar{\lambda}\} \cup (\bigcup_{i\in I}(\lambda_i \setminus (\bigcup_{\bar{\tau}\in\bar{\lambda}}\tau_i(\bar{\tau}))))$$

Then every τ in λ is a track on J and any two tracks in λ do not intersect. Moreover, for every track τ in λ, we have either $\tau\cap J_i=\tau_i(\tau\cap\bar{J})$ or $\tau\cap J_i=\emptyset$, i.e., $\lambda|J_i=\lambda_i$ for all i in I. Hence the hypothesis pair $(\lambda_{i_1},\lambda_{i_2})$ is fusable. When $\#(I)>2$, we must construct hypotheses similarly but we must start with the common predecessor tracks in $\bigcap_{i\in I}J_i$ and repeat the process for all the intersections with gradually fewer terms. *Q.E.D.*

Remark: Since $J\subseteq\mathbf{J}$ implies $\tau\cap J\in\mathbf{T}(J)$ for any track $\tau\in\mathbf{T}(\mathbf{J})$, the restriction operation (40) defined a mapping from $\mathbf{T}(\bigcup_{i\in I}J_i)$ to $\prod_{i\in I}\mathbf{T}(J_i)$. The definition of the fusability implies that the range of this mapping is exactly the set of all the fusable tuples of tracks. Suppose $\tau\in\mathbf{T}(\bigcup_{i\in I}J_i)$ and $\tau\cap J_i=\tau_i$ for all i in I. Then, $\bigcup_{i\in I}\tau_i=\bigcup_{i\in I}(\tau\cap J_i)=\tau\cap(\bigcup_{i\in I}J_i)=\tau$. Therefore, the unionization operation is indeed the inverse mapping, and hence, the set of all the tracks on $\bigcup_{i\in I}J_i$, i.e., $\mathbf{T}(\bigcup_{i\in I}J_i)$, is isomorphic to the set of all the fusable tuples of tracks. Similarly $J\subseteq\mathbf{J}$ implies $(\lambda\mid J)\in\mathbf{H}(J)$ for any hypothesis $\lambda\in\mathbf{H}(\bigcup_{i\in I}J_i)$. Therefore the restriction operation (41) defines a surjective mapping from $\mathbf{H}(\bigcup_{i\in I}J_i)$ to the set of all the fusable hypothesis tuples. This mapping, however, may not be bijective (one-to-one). A counterexample was given in [1].

Lemma 5 provides us with ways of generating $\mathbf{T}(\bigcup_{i\in I}J_i)$ and $\mathbf{H}(\bigcup_{i\in I}J_i)$ from $\prod_{i\in I}\mathbf{T}(J_i)$ and $\prod_{i\in I}\mathbf{H}(J_i)$, respectively. However, checking the conditions described

by (42) or (43) may not be directly conducted on the information nodes which precedes I. Nonetheless, by applying *Lemma 2* in Section 1, we can transform (42) and (43) into a form by which predecessor-consistency tests are required only on certain predecessor nodes of I. The result is stated as follows:

Theorem 4: Let i_0 be a communication receiving node and I be the set of all the immediate predecessors of it. For each $(i_1,i_2)\in I\times I$, let $\bar{I}(i_1,i_2)$ be a set of information nodes $\bar{i}$ such that $\bar{i}<i_1$ and $\bar{i}<i_2$. Then, we have
(1) *a necessary and sufficient condition for any track tuple $(\tau_i)_{i\in I}\in\prod_{i\in I}\mathbf{T}(J^{(i)})$ to be fusable is that, for any $(i_1,i_2)\in I\times I$,*

$$\tau_{i_1}\bigcap J^{(\bar{i})}=\tau_{i_2}\bigcap J^{(\bar{i})} \tag{45}$$

for any $\bar{i}\in\bar{I}(i_1,i_2)$, and
(2) *a necessary condition for any hypothesis tuple $(\lambda_i)_{i\in I}\in\prod_{i\in I}\mathbf{H}(J^{(i)})$ to be fusable is that, for any $(i_1,i_2)\in I\times I$,*

$$\lambda_{i_1}\mid J^{(\bar{i})}=\lambda_{i_2}\mid J^{(\bar{i})} \tag{46}$$

for any $\bar{i}\in\bar{I}(i_1,i_2)$.

Proof: Suppose $(i_1,i_2)\in I\times I$. Then, by *Lemma 2* of Section 1, we have

$$\bigcup_{i\in\bar{I}(i_1,i_2)}J^{(\bar{i})}=J^{(i_1)}\bigcap J^{(i_2)}. \tag{47}$$

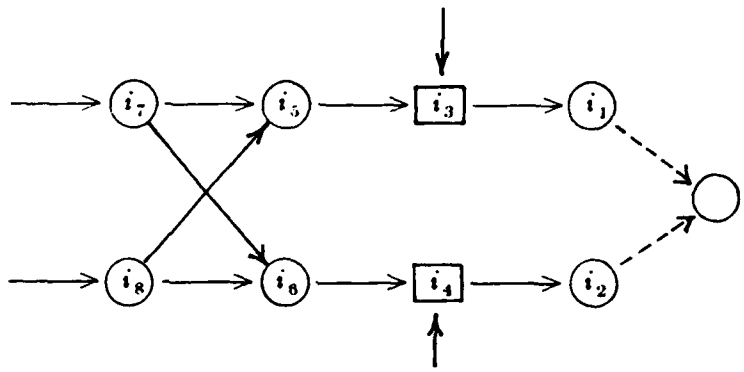
Therefore, if (45) holds for all $\bar{i}$ in $\bar{I}(i_1,i_2)$, (42) holds. Conversely, for any $\bar{i}$ in $\bar{I}(i_1,i_2)$, $J^{(\bar{i})}\subseteq J^{(i_1)}\bigcap J^{(i_2)}$, and hence, (42) implies (45) and (43) implies (46). *Q.E.D.*

Remarks:
(1) For each $(i_1,i_2)\in I\times I$ such that $i_1\neq i_2$, a set $\bar{I}(i_1,i_2)$ of common predecessors $\bar{i}$ such that $\bar{i}<i_1$ and $\bar{i}<i_2$ may not be unique. When the above theorem is used to test fusability, any information node $\bar{i}$ in $\bar{I}(i_1,i_2)$ such that $\bar{i}<\bar{i}'$ with another

$\bar{i}'$ in $\bar{I}(i_1,i_2)$ can be excluded from $\bar{I}(i_1,i_2)$.

(2) The type of tests defined by (45) or (46) provides a necessary and sufficient condition for track fusability but only a necessary condition for hypothesis fusability. This is due to the fact that a fusable tuple of tracks produces only one fused track but a fusable tuple of hypotheses may produce more than one hypotheses. Thus, (47) is not a sufficient condition for the hypothesis fusability as shown below:

Counterexample:



This is an example of broadcasting type communication and we have $J^{(i_5)}=J^{(i_6)}=J^{(i_7)}\cup J^{(i_8)}=J^{(i_1)}\cap J^{(i_2)}$. Let (λ_1,λ_2) be an arbitrary pair of hypotheses on $J^{(i_1)}$ and $J^{(i_2)}$, respectively, to be fused. They may share identical predecessors $\lambda_7\in\mathbf{H}(J^{(i_7)})$ and $\lambda_8\in\mathbf{H}(J^{(i_8)})$ while predecessor λ_5 on $J^{(i_5)}$ of λ_1 may differs from the predecessor λ_6 on $J^{(i_6)}$ of λ_2. In such a case, λ_1 and λ_2 cannot be fused.

By the fact stated in (2) of the above remarks, we cannot determine the fusability of hypothesis tuples by this theorem. However, part (1) of this theorem can be used indirectly to determine the fusability of hypothesis tuples according to the theorem stated below. On the other hand, the above theorem can be used

to screen hypothesis tuples for candidates to be fused.

Theorem 5: Let $(J_i)_{i\in I}$ *be any tuple of measurement index sets and* $J=\bigcup_{i\in I} J_i$. *Then, for any given* $\lambda\subseteq \mathbf{T}(J)\setminus\{\emptyset\}$ *and any given* $(\lambda_i)_{i\in I}\in\prod_{i\in I}\mathbf{H}(J_i)$, *we have* $\lambda\in\mathbf{H}(J)$ *and* $\lambda \mid J_i=\lambda_i$ *for all* i *in* I *if and only if*

(1) *for any* τ *in* λ, *there exists a fusable track tuple* $(\tau_i)_{i\in I}\in\prod_{i\in I}(\lambda_i\cup\{\emptyset\})$ *such that* $\tau=\bigcup_{i\in I}\tau_i$, *and*

(2) *for all* $i\in I$ *and and for all* $\tau_i\in\lambda_i$, *there exists a unique* τ *in* λ *such that* $\tau_i\subseteq\tau$.

Proof: [if] Assume (1) and (2) hold for given $\lambda\subseteq\mathbf{T}(J)\setminus\{\emptyset\}$ and $(\lambda_i)_{i\in I}\in\prod_{i\in I}\mathbf{H}(J_i)$. To show $\lambda\in\mathbf{H}(J)$ it suffices to show λ does not have any overlapping tracks. Suppose there are overlapping tracks τ_1 and τ_2 in λ, i.e., $(\tau_1,\tau_2)\in\lambda\times\lambda$, $\tau_1\neq\tau_2$ and $\tau_1\cap\tau_2\neq\emptyset$. Then by (1), for each $l\in\{1,2\}$ there exists a fusable track tuple $(\tau_{li})_{i\in I}\in\prod_{i\in I}(\lambda_i\cup\{\emptyset\})$ such that $\tau_l=\bigcup_{i\in I}\tau_{li}$. Since $\tau_1\cap\tau_2\neq\emptyset$, there exists a $(i_1,i_2)\in I\times I$ such that $\tau_{1i_1}\cap\tau_{2i_2}\neq\emptyset$. For each $l\in\{1,2\}$, $\tau_{li_l}\in\lambda_{i_l}$ because $\tau_{li_l}\neq\emptyset$. Since $\lambda_{i_l}\in\mathbf{H}(J_{i_l})$ for each l, $i_1=i_2$ implies $\tau_{1i_1}=\tau_{2i_2}$. If we have $\tau_{1i_1}\subseteq\tau_1$ and $\tau_{1i_1}\subseteq\tau_2$, the uniqueness part of (2) is violated. Therefore, we must have $i_1\neq i_2$. On the other hand, the fusability assumption in (1) implies $\tau_{li}=\tau_l\cap J_i$ for each $(l,i)\in\{1,2\}\times I$. Hence we have

$$\emptyset\neq\tau_{1i_1}\cap\tau_{2i_2}=(\tau_1\cap J_{i_1})\cap(\tau_2\cap J_{i_2})\subseteq\tau_2\cap J_{i_1}\cap J_{i_2}=\tau_{2i_1}\cap J_{i_2}\subseteq\tau_{2i_1}$$

which means $\tau_{1i_1}\cap\tau_{2i_1}\neq\emptyset$. Also the uniqueness part of (2) prohibits $\tau_{1i_1}=\tau_{2i_1}$. This contradicts with $\lambda_{i_1}\in\mathbf{H}(J_{i_1})$, $\tau_{1i_1}\in\lambda_{i_1}$ and $\tau_{2i_1}\in\lambda_{i_1}$. Therefore, λ must be a hypothesis on J, or $\lambda\in\mathbf{H}(J)$.

Let $i\in I$. Suppose $\tau_i\in\lambda_i$. There exists a $\tau\in\lambda$ such that $\tau_i\subseteq\tau$ by (2). Then, by (1), there exists a fusable track tuple $(\tau'_{i'})_{i'\in I}\in\prod_{i'\in I}(\lambda_{i'}\cup\{\emptyset\})$ such that $\tau=\bigcup_{i'\in I}\tau'_{i'}$. Then, for each i, the fusability implies $\emptyset\neq\tau_i\subseteq\tau\cap J_i=\tau'_i$. Since $\lambda_i\in\mathbf{H}(J_i)$, we

must have $\tau'_i = \tau_i$, i.e., $\tau_i \in \lambda|J_i$. On the other hand, if $\tau_i \in \lambda|J_i$, i.e., if $\tau_i \neq \emptyset$ and $\tau \cap J_i = \tau_i$, (1) and (2) similarly imply $\tau_i \in \lambda_i$. Hence we have $\lambda|J_i = \lambda_i$.

[only if] Suppose $\lambda \in \mathbf{H}(J)$ and $\lambda|J_i = \lambda_i$ for all $i \in I$. For all $\tau \in \lambda$ and for all $i \in I$, let $\tau_i = \tau \cap J_i$. Then, $(\tau_i)_{i \in I}$ apparently satisfies the requirements of (1). (2) follows directly from $\lambda|J_i = \lambda_i$. λ is a hypothesis on J, and hence, does not have any overlapping tracks. This proves the uniqueness part. *Q.E.D.*

Remark: From the proof it is clear that the condition, $\tau_i \subseteq \tau$ in (2) of *Theorem 5* can be replaced by $\tau_i = \tau \cap J_i$.

Hypothesis Evaluation: In addition to *Assumptions* [A1] - [A8], [C1] - [C3] and the assumptions made earlier in Section 3.2, we have to add the following:

[E1] (Poisson-i.i.d. Targets) The number N_T of targets is constant over time and has a Poisson distribution with mean $\nu_0 > 0$. Given $N_T > 0$, $(x_i(t))_{i=1}^{N_T}$ is a system of independent Markov processes sharing the identical joint distributions. Moreover, each process is bi-directionally deterministic having an identical group $(\Phi_{\Delta t})_{\Delta t=-\infty}^{\infty}$ of deterministic state transitions and an identical initial state distribution which has density $q_{t_0}(x(t_0))$.

[E2] (Independent Detection) For each $k = (s,t) \in \mathbf{K}$, we have[1]

$$Prob\,(Dom\,(A(k)) \mid (x_i(t))_{i=1}^{N_T}, N_T) = \tag{48}$$

$$\prod_{i=1}^{N_T} p_D(x_i(t) \mid k)^{\chi(i;Dom(A(k)))} (1 - p_D(x_i(t) \mid k))^{(1-\chi(i;Dom(A(k))))}$$

[E3] (Random Assignment) For any $k \in \mathbf{K}$, given $N_M(k)$ and $Dom\,(A(k))$, every realization of $A(k)$ is equally probable.

[E4] (False Alarm Number) For each $k = (s,t) \in \mathbf{K}$, we have

---

1 For any function $f : E \to F$, $Dom\,(f)$ is its domain and $Im\,(f)$ is its range. $\chi(\cdot;E)$ is the indicator function of set E, i.e., $\chi(e;E)$ is 1 if $e \in E$ 0 otherwise.

$$Prob\,(N_M(k)\mid A(k),(z_i(t))_{i=1}^{N_T},N_T) = p_{N_{FA}}(N_M(k) - \#\,(Im\,(A(k)))\mid k) \qquad (49)$$

[E5] (Independent Measurement) For each $k=(s,t)\in\mathbf{K}$ and $z_k=((y_j(k))_{j=1}^{N_M(k)},N_M(k),k)\in Z$, we have

$$Prob\,((y_j(t))_{j=1}^{N_M(k)}\in(dy_j)_{j=1}^{N_M(k)}\mid A(k),N_M(k),(x_i)_{i=1}^{N_T},N_T) = \qquad (50)$$

$$\Big(\prod_{i\in Dom(A)} p_M(y_{A(i)}\mid x_i(t))\mu_s(dy_{A(i)})\Big)\cdot\Big(\prod_{j\in J_{FA}(k)} p_{FA}(y_j\mid k)\mu_s(y_j)\Big)$$

where

$$J_{FA}(k) = \{1,\ldots..N_M(k)\}\backslash Im\,(A(k)) \qquad (51)$$

For any track $\tau\in\mathbf{T}(\mathbf{J})$ and any information set Z with index set K and measurement index set J, define *track likelihood* of track τ given Z by

$$l(\tau,Z) = \nu_0\, c(\tau,Z) \qquad (52)$$

where

$$c(\tau,Z) = \int\Big(\prod_{k\in K} g(y(\tau\mid k)\mid \Phi_{t_k-t_0}(x),k)\Big)\, q_{t_0}(x)\mu(dx) \quad , \qquad (53)$$

with $Z=\{((y_j(k))_{j=1}^{N_M(k)},N_M(k),s_k,t_k)\mid k\in K\}$,

$$y(\tau\mid k) = \begin{cases} y_j(k) & \text{if } (j,k)=(\tau\mid k)\neq\emptyset \\ \emptyset & otherwise \end{cases} \qquad (54)$$

and

$$g(y \mid x,k) = \begin{cases} p_M(y \mid x,k)\, p_D(x \mid k) & \text{if } y \neq \emptyset \\ 1-p_D(x \mid k) & \text{otherwise} \end{cases} \tag{55}$$

For any $t \in T$, let the measure σ_t on X be defined by

$$\sigma_t(E) = \int_{(\Phi_t \, t_0)^{-1}(E)} q_{t_0}(x)\mu(dx) \tag{56}$$

and let $q_t(\cdot)$ be the density of σ_t with respect to μ. Then we have the following theorem:

Theorem 6: Let Z be any information set, and let K and J be the corresponding index set and measurement index set. Then we have

(1)

$$Prob(\Lambda|J \mid Z) = P(Z)^{-1} \exp(l(\emptyset,Z)-\nu_0) \tag{57}$$

$$\prod_{k \in K} \left(\frac{N_{FA}(k)!}{N_M(k)!} p_{N_{FA}}(N_{FA}(k) \mid k) \prod_{j \in J_{FA}(k)} p_{FA}(y_j \mid k) \right) \cdot \prod_{\tau \in \Psi} l(\tau,Z)$$

where $N_{FA}(k) = N_M(k) - \#\left(\bigcup_{\tau \in \Psi} \tau \mid k\right)$.

(2)

$$\nu(K) \triangleq E(N_T - \#(\Lambda|J) \mid \Lambda|J,K) = l(\emptyset,Z) \tag{58}$$

(3) For any $t \in T$,

$$Prob\left\{(x_i(t))_{i=1}^{N_T} \in (dx_i)_{i=1}^{N_T} \mid \Omega, N_T, \Lambda|J, Z\right\} = \tag{59}$$

$$\left(\prod_{\tau \in \Psi} p_t(x_{\Omega(\tau)} \mid \tau, Z)\mu(dx_{\Omega(\tau)})\right)\left(\prod_{i \notin Im(\Omega)} p_t(x_i \mid \emptyset, Z)\mu(dx_i)\right)$$

where $\Omega: \Lambda|J \to I_T$ is one of the equally probable 1-to-1 assignment functions from tracks to target indices, and

$$p_t(x \mid \tau, Z) = c(\tau, Z)^{-1} \left(\prod_{k \in K} g(y[\tau \mid k] \mid \Phi_{t_k - t}(x), k) \right) q_t(x) \quad . \tag{60}$$

Proof: Suppose K is ordered arbitrarily as $K = \{k_1, k_2, \ldots, k_m\}$ such that $t_{k_i} \leq t_{k_{i+1}}$. Then, according to *Corollary to Theorem 2* in Section 5 of [2], we have (57) with

$$l(\tau, Z) = \nu_0 \prod_{i=1}^{m} \int g(y[\tau \mid k_i] \mid x, k_i) p_{t_{k_i}}(x \mid \tau \cap J_{i-1}, Z_{i-1}) \mu(dx) \tag{61}$$

where $J_i = \{(j, k_{i'}) \in J \mid i' \leq i\}$ and $Z_i = \{(\zeta, k_{i'}) \in Z \mid i' \leq i\}$. By means of mathematical induction, we can show

$$\begin{aligned} &\prod_{i=1}^{m} \int g(y[\tau \mid k_i] \mid x, k_i) p_{t_{k_i}}(x \mid \tau \cap J_{i-1}, Z_{i-1}) \mu(dx) \qquad (62) \\ &= \int \prod_{i=1}^{m} g(y[\tau \mid k_i] \mid \Phi_{t_{k_i} - t_0}(x), k_i) q_{t_0}(x) \mu(dx) \\ &= \int \prod_{i=1}^{m} g(y[\tau \mid k_i] \mid \Phi_{t_{k_i} - t}(x), k_i) q_t(x) \mu(dx) \end{aligned}$$

Thus Part (1) follows from (52) - (56) and (61) - (62).

According to Part [ii] of *Theorem 2* in Section 5 of [2], with the same ordering of K as above, we have

$$\nu_{k_i} = \nu_{k_{i-1}} \int g(\theta \mid x, k_i) p_{t_{k_i}}(x \mid \emptyset, Z_{i-1}) \mu(dx) \quad . \tag{63}$$

By repeatedly applying (63), we obtain Part (2). Part (3) then follows from *Lemma 2* and Part [i] of *Theorem 2* of Section 5 in [2]. *Q.E.D.*

The distributed version of this theorem follows:

Theorem 7: Let $i_0 = (t, n, CR)$ be a communication receiving node in $\mathbf{I}_{CR}$ and I be the set of all the immediate predecessors of i_0. Let $Z = \bigcup_{i \in I} Z^{(i)}$ with K and J be the associating index set and measurement index set. Let $(\bar{T}, \alpha)$ be the

pair which satisfies the conditions (24) *and* (25) *of Theorem 3. Then we have*

(1)

$$Prob\,(\Lambda|J \mid Z) = d \prod_{\bar{i}\in\bar{I}} Prob\,(\Lambda|J^{(\bar{i})} \mid Z^{(\bar{i})})^{\alpha(\bar{i})} \prod_{\tau\in\Psi} \tilde{l}\,(\tau,(Z^{(\bar{i})})_{\bar{i}\in\bar{I}}) \tag{64}$$

where

$$d = P(Z)^{-1} \left(\prod_{\bar{i}\in\bar{I}} P(Z^{(\bar{i})})^{\alpha(\bar{i})}\right) \exp\left(\nu(K) - \sum_{\bar{i}\in\bar{I}} \alpha(\bar{i})\nu(K^{(\bar{i})})\right) , \tag{65}$$

$$\tilde{l}\,(\tau,(Z^{(\bar{i})})_{\bar{i}\in\bar{I}}) = \left(\prod_{\bar{i}\in\bar{I}} \nu(K^{(\bar{i})})^{\epsilon_{\bar{i}}(\tau)\alpha(\bar{i})}\right) \tilde{c}\,(\tau,(Z^{(\bar{i})})_{\bar{i}\in\bar{I}}) \tag{66}$$

$$\epsilon_{\bar{i}}(\tau) = \begin{cases} 1 & \text{if } \tau\cap J^{(\bar{i})}=\emptyset \\ 0 & \textit{otherwise} \end{cases} , \tag{67}$$

and

$$\tilde{c}\,(\tau,(Z^{(\bar{i})})_{\bar{i}\in\bar{I}}) = \int \prod_{\bar{i}\in\bar{I}} p_t(x \mid (\tau\cap J^{(\bar{i})}),Z^{(\bar{i})})^{\alpha(\bar{i})} \mu(dx) , \tag{68}$$

(2)

$$p_t(x \mid \tau,Z) = \tilde{c}\,(\tau,(Z^{(\bar{i})})_{\bar{i}\in\bar{I}})^{-1} \prod_{\bar{i}\in\bar{I}} p_t(x \mid (\tau\cap J^{(\bar{i})}),Z^{(\bar{i})})^{\alpha(\bar{i})} , \tag{69}$$

and

(3)

$$\nu(K) = \tilde{l}\,(\emptyset,(Z^{(\bar{i})})_{\bar{i}\in\bar{I}}) = \left(\prod_{\bar{i}\in\bar{I}} \nu(K^{(\bar{i})})^{\alpha(\bar{i})}\right) \int \prod_{\bar{i}\in\bar{I}} p_t(x \mid \emptyset,Z^{(\bar{i})})^{\alpha(\bar{i})} \mu(dx) \tag{70}$$

Proof: Let $(\bar{I},\alpha)$ be the pair which satisfies the conditions (24) and (25). Then we should first note that, by *Lemma 3* and *Theorems 1* and *2*, we have

$$a[K] \triangleq \prod_{k\in K} a_k = \prod_{\bar{i}\in\bar{I}}\Big(\prod_{k\in K^{(\bar{i})}} a_k\Big)^{\alpha(\bar{i})} = \prod_{\bar{i}\in\bar{I}}\ \prod_{k\in K^{(\bar{i})}} (a_k)^{\alpha(\bar{i})} \tag{71}$$

for any $(a_k)_{k\in K}$ such that $a_k > 0$ for all $k\in K$. Also, as shown in the proof of *Theorem 1*, if a_k is constant, i.e., if $a_k = \bar{a}$ for all $k\in K$, we have $a[K] = \bar{a}$. For notational simplicity, we abreviate notations as $g_k(x,\tau) = g(y[\tau[k]\,|\,\Phi_{t_i-t}(x),k)$ and

$$l_k^{FA} = \frac{N_{FA}(k)!}{N_M(k)!}\, p_{N_{FA}}(N_{FA}(k)\,|\,k) \prod_{j\in J_{FA}(k)} p_{FA}(y_j(k)\,|\,k)$$

Then, it follows from Part (1) of *Theorem 6* that

$$\prod_{\bar{i}\in\bar{I}} Prob\,(\Lambda[J^{(\bar{i})}\,|\,Z^{(\bar{i})}) \tag{72}$$

$$= \prod_{\bar{i}\in\bar{I}}\Big(P(Z^{(\bar{i})})^{-1}\exp\big(l(\emptyset,Z^{(\bar{i})})-\nu_0\big)\Big(\prod_{k\in K^{(\bar{i})}} l_k^{FA}\Big)\Big(\prod_{\tau\in\Psi^{(\bar{i})}} l(\tau,Z^{(\bar{i})})\Big)\Big)^{\alpha(\bar{i})}$$

$$= \Big(\prod_{\bar{i}\in\bar{I}} P(Z^{(\bar{i})})^{-\alpha(\bar{i})}\Big)\exp\Big(\sum_{\bar{i}\in\bar{I}}\alpha(\bar{i})\,l(\emptyset,Z^{(\bar{i})})\Big)\exp(-\nu_0)$$

$$\Big(\prod_{\bar{i}\in\bar{I}}\ \prod_{k\in K^{(\bar{i})}} (l_k^{FA})^{\alpha(\bar{i})}\Big)\Big(\prod_{\bar{i}\in\bar{I}}\ \prod_{\tau\in\Psi^{(\bar{i})}} l(\tau,Z^{(\bar{i})})\Big)$$

It follows from (71) that

$$\prod_{k\in K} l_k^{FA} = \prod_{\bar{i}\in\bar{I}}\ \prod_{k\in K^{(\bar{i})}} (l_k^{FA})^{\alpha(\bar{i})} \tag{73}$$

and

$$\Big(\prod_{k\in K} g_k(x,\tau)\Big)q_t(x) = \prod_{\bar{i}\in\bar{I}}\ \prod_{k\in K^{(\bar{i})}} \big(g_k(x,\tau)q_t(x)\big)^{\alpha(\bar{i})}\ . \tag{74}$$

On the other hand, it follows from (60) that

$$p_t(x\ |\ \tau\cap J^{(\bar{i})}, Z^{(\bar{i})})^{\alpha(\bar{i})} = c\,(\tau\cap J^{(\bar{i})}, Z^{(\bar{i})})^{-\alpha(\bar{i})} \prod_{k\in K^{(\bar{i})}} \big(g_k(x,\tau\cap J^{(\bar{i})})q_t(x)\big)^{\alpha(\bar{i})} \tag{75}$$

for each $\bar{i}\in\bar{I}$. Then it follows from (52), (53), (74) and (75) that

$$l(\tau,Z) = \Big(\prod_{\bar{\imath}\in\bar{I}} l(\tau\cap J^{(\bar{\imath})},Z^{(\bar{\imath})})^{\alpha(\bar{\imath})}\Big)\Big(\int \prod_{\bar{\imath}\in\bar{I}} p_t(x \mid \tau\cap J^{(\bar{\imath})},Z^{(\bar{\imath})})^{\alpha(\bar{\imath})}\mu(dx)\Big) \qquad (76)$$

Part **(1)** then follows from (72), (73) and (76). Parts **(2)** and **(3)** cab be proved similarly. *Q.E.D.*

Remark: By the definition of $\tilde{l}$ by (66) - (67), we can interpret $\tilde{l}$ as *track-to-track likelihood* of the tuple $(\tau\cap J^{(\bar{\imath})})_{\bar{\imath}\in\bar{I}}$ of tracks. This likelihood can also be defined as

$$\tilde{l}(\tau,(Z^{(\bar{\imath})})_{\bar{\imath}\in\bar{I}}) = \int \prod_{\bar{\imath}\in\bar{I}} \tilde{p}_t(x \mid (\tau\cap J^{(\bar{\imath})}),Z^{(\bar{\imath})})^{\alpha(\bar{\imath})}\mu(dx) \qquad (77)$$

where

$$\tilde{p}_t(x \mid (\tau\cap J^{(\bar{\imath})}),Z^{(\bar{\imath})}) = \nu(K^{(\bar{\imath})})^{\in_{\bar{\imath}}(\tau\cap J^{(\bar{\imath})})}\, p_t(x \mid (\tau\cap J^{(\bar{\imath})}),Z^{(\bar{\imath})}) \qquad (78)$$

4. Conclusion

Solutions were given to distributed estimation problems and to distributed multitarget tracking problems with arbitrary communication patterns defined by information graphs in bi-directionally diterministic cases. It is expected that the results described in this appendix provide sufficiently functioning distributed algorithms even without the bi-directional determinancy if the randomness in state transition is small and the communication is reasonably frequent.